AN IMPLEMENTATION OF AN
ITERATIVE GLOBAL
FLOW ANALYSIS ALGORITHM.

Jack William Cowan

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN IMPLEMENTATION OF AN
ITERATIVE GLOBAL
FLOW ANALYSIS ALGORITHM

by

Jack William Cowan

March 1975

Thesis Advisor: G. A. Kildall

Approved for public release; distribution unlimited.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) AN IMPLEMENTATION OF AN ITERATIVE GLOBAL FLOW ANALYSIS ALGORITHM	5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; March 1975
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Jack W. Cowan	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	12. REPORT DATE March 1975
	13. NUMBER OF PAGES 290
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Naval Postgraduate School Monterey, California 93940	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Global Optimization, Program Flow Graph, Basic Block, Meet Operator, Code Synthesis Filter.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Kildall has stated a general data flow analysis algorithm which has been applied to several forms of classical global program optimization. The algorithm operates upon the flow graph of a program, where the nodes correspond to basic blocks and the edges represent possible program control flows. In order to test the effectiveness of this algorithm, a general purpose optimizing module was written in XPL which analyzes

ALGOL-E programs for constant computations, common subexpressions and simplifying formal identities. Various node selection algorithms were investigated with respect to the convergence rate of the algorithm.

An Implementation of an
Iterative Global
Flow Analysis Algorithm

by

Jack William Cowan
Lieutenant(JG), United States Navy
B.S., Naval Postgraduate School, 1974

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 1975

ABSTRACT

Kildall has stated a general data flow analysis algorithm which has been applied to several forms of classical global program optimization. The algorithm operates upon the flow graph of a program, where the nodes correspond to basic blocks and the edges represent possible program control flows. In order to test the effectiveness of this algorithm, a general purpose optimizing module was written in XPL which analyzes ALGOL-E programs for constant computations, common subexpressions and simplifying formal identities. Various node selection algorithms were investigated with respect to the convergence rate of the algorithm.

TABLE OF CONTENTS

TABLE OF CONTENTS (CON'T)

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENT

Without the patience and assistance of the Computer Operations staff this project would have been much less enjoyable. In particular, the efficient and reliable operating procedures of Edwin Donnellan, Mannas Anderson and Kristina Butler are deeply appreciated. A word of appreciation must also be extended to Katheryn Strutynski for her assistance with enumerable system problems. Finally, I am grateful to my wife, Patricia, for her many long and arduous hours keypunching and in obtaining needed reference materials. Without her willing and able assistance this project would never have been concluded.

I. INTRODUCTION

A. BACKGROUND

The increased use of high level language compilers has generated an interest in producing efficient machine language code from the high level source language. Generally, this interest has resulted in techniques for transforming the original source program to produce an equivalent form which is optimized with respect to some criteria.

Machine independent optimization algorithms can be divided into two broad subclasses - local and global. Basically, the difference is that global techniques take into account the overall flow of the program to be optimized while local methods do not. Specifically, local techniques require that the program to be optimized be partitioned into "basic blocks." These blocks are divisions of the program which have no transfers into or out of the block except for a transfer into an initial element and a transfer out of the last element. Each block is then optimized independent of the other existing blocks.

A number of techniques have evolved for analyzing global program structure, including recent work by Kildall [30], Hecht and Ullman [22], Allen [4], Kennedy [25], Graham and Wegman [20]. These techniques analyze global flow by representing the program as a directed graph with the nodes of the graph corresponding to basic blocks with the edges showing possible program control flows. If the ALGOL program of Figure 1 is analyzed for redundant calculations, local techniques would not detect the obvious redundancy of (C + B) in block D because this occurrence of the expression

resides in a different block than the previous calculation of (B + C). On the other hand, global flow analysis would detect the redundant calculation since (B + C) has been computed on all paths to block D.

An iterative global flow analysis algorithm was developed by Kildall [28] that is capable of analyzing any program structure. The algorithm propagates data from block to block until the propagated data reaches a terminal state. This process may necessitate multiple passes through some blocks. Kildall has shown that the algorithm is finite and will succeed given any order for processing the basic blocks. However, using a simulation, Lukasczyk [34] has shown that the convergence rate of the algorithm can be affected by the order in which the "basic blocks" are processed. As one would expect, his simulation further indicates that as the number of nodes in a program flow graph increases, the method of block selection becomes more critical.

An implementation of the global flow analysis algorithm to be presented was originally done by Kildall [30]. Various improvements in the implementation were made, including more complete data structures for computing flow information. The results of Lukasczyk's simulation was then compared to the actual results from the implementation.

B. DEFINITIONS

In order to examine the global flow analysis algorithm stated by Kildall [30], various basic notions need be presented.

```
IF X > Y THEN
BEGIN
  A := B + C;
  Y := X;
END
ELSE
  H := B + C;
R := C + B;
```

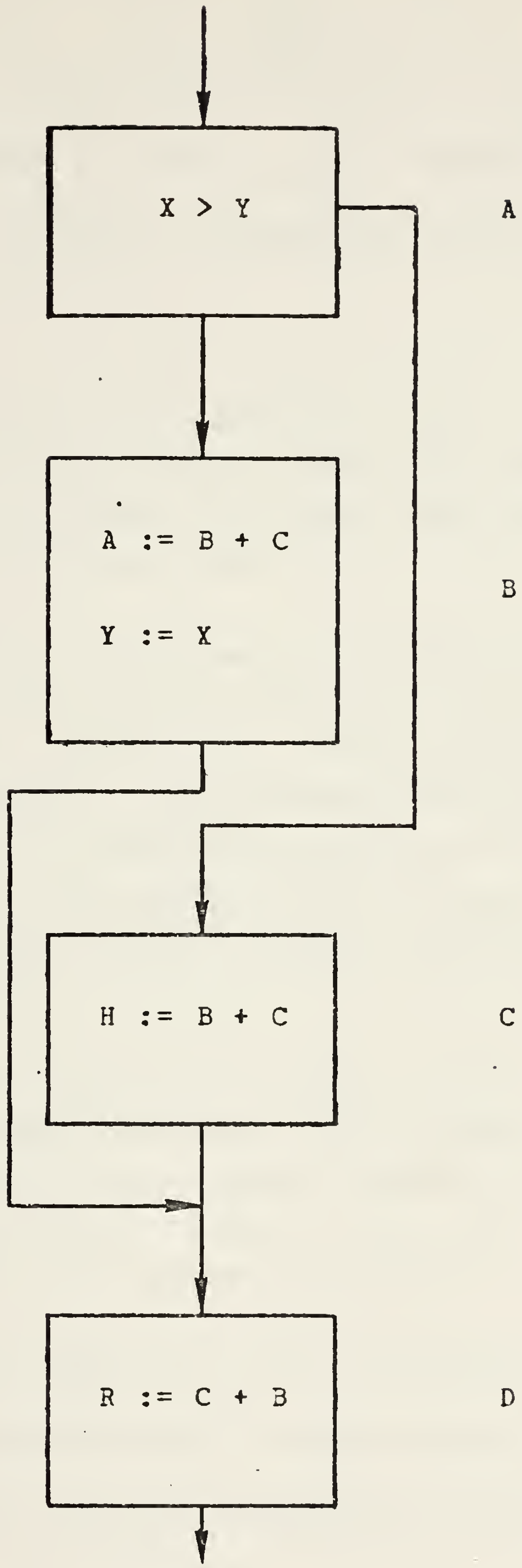

FIGURE 1

AN ALGOL PROGRAM AND ITS FLOWGRAPH

A _directed graph_ _G_ is a two-tuple (N, E), where N represents the set of "nodes" and E is a subset of N x N called "edges." Given an ordered pair of nodes (n_1, n_2) in E, an edge is said to "leave" node n_1 and "enter" node n_2. Further, n_1 is an "immediate predecessor" of n_2, and n_2 is an "immediate successor" of n_1.

A _program flow graph_ _G_ is a three-tuple (N, E, R), where N and E are as above and R is a non-empty subset of N containing the entry nodes of G such that there exists a path from an entry node to any node in N.

As stated previously, a program can be partitioned into a set of _basic blocks_. Each basic block is a program segment having no transfers entering or exiting except thru the initial and final elements. Basic blocks are represented by single nodes in the program flow graph and are hereafter referred to as blocks. The edges of the program flow graph represent the possible paths of control flow as directed by the program structure.

In the following discussion, note that for simplicity the various pools of information are always associated with the node "n." Where this rather informal association becomes unclear, particular effort will be made to distinguish pools with their associated nodes.

A _current optimizing pool_ (P_c) is associated with a particular block n and represents the optimizing information available at the block, which is iteratively refined during the flow analysis.

An _input pool_ (P_i) is information which is to be used upon entry to a given block n. The optimizing function, defined below, uses this pool in its transformation. This information is identical to the current optimizing pool of a block n, and will be referred to as the input pool to increase the clarity of the global flow analysis algorithm which is stated below.

An _output pool_ (P_o) is optimizing information created by the application of the optimizing function to an input pool for a block. Upon exiting a block, this set is present and is used in performing the meet operator which is defined below.

A _final optimized pool_ (P_f) is the final optimizing information that is distinguished as the meet over all paths for a block n. This pool is produced by the global flow analysis algorithm for each block.

A _meet operator_ is a binary operation denoted $\wedge$, on P x P to P, where P is the set of all possible optimizing pools. The operator combines an output pool (P_o) for a block n and a current optimized pool ($P_c{}'$) of an immediate successor block n' to produce an input pool ($P_i{}'$) for n'.

$$P_i{}' \leftarrow P_o \wedge P_c{}'$$

The following properties must hold for all a, b and c in P:

1) $a \wedge a = a$ (idempotent)

2) $a \wedge b = b \wedge a$ (communitive)

3) $a \wedge (b \wedge c) = (a \wedge b) \wedge c$ (associative)

The meet operation results in a partial ordering upon the optimization elements: $a \le b$ if and only if $a \quad b = b$.

An *optimizing function* maps an input pool to an output pool and must have the homomorphism property:

$$f(n, a \wedge b) = f(n, a) \wedge f(n, b)$$

for all nodes n in N and input pools a and b in P. P is assumed to contain an identity element such that

$$1 \wedge a = a \text{ for all } a \text{ in } P.$$

C. A GLOBAL FLOW ANALYSIS ALGORITHM

Although a basic familiarization with Kildall's flow analysis algorithm [28] is assumed, some fundamental notions are presented below for completeness.

The global flow analysis algorithm, Algorithm Q, of Kildall was developed for use in the compile time optimization of object code. Kildall [30] has pointed out that many techniques have been developed to optimize straight line code while others take into account program flow branching. The Algorithm Q was designed to permit extensions of straight line optimization techniques to the global case.

The Algorithm Q is iterative and may be used on any program flow graph. As stated previously, the algorithm propagates data from block to block until the propagated data reaches a final state for a given block. This final state is the "meet over all paths," called the MOP solution, for a specified block and provides information with which the block can be optimized.

Informally the Algorithm Q can be stated as:

Select a node from the list of nodes to be processed, its current pool will be used as input.

Apply the optimizing function to the input pool and produce an output pool.

For each immediate successor, apply the meet operation on the output pool and the successor's current pool. If the result is strictly smaller than the successor's current pool, then reset the current pool to this result and add the successor to the nodes to be processed list.

A particular form of the Algorithm Q which is appropriate for machine implementation is stated below.

Q1 [initialize]	initialize the investigation list L to the entry node $L \leftarrow \{n\}$ and set the current pool of n to empty. $P_c \leftarrow 0$
Q2 [terminate?]	if L is empty then halt, otherwise
Q3 [select a node]	let n be an element of L, and set $L \leftarrow L - \{n\}$.
Q4 [apply function]	apply the optimizing function to the P_c of n producing

output information P_o for the node n.

Q5 [enter nodes] for each n' which is an immediate successor of n, form $P_i' \leftarrow P_o \wedge P_c'$ where P_i' and P_c' correspond to node n'. If $P_i' \leq P_c'$ and $P_i' \neq P_c'$ then set

$$P_c' \leftarrow P_i'$$

and

$$L \leftarrow L \cup \{n'\}.$$

Q6 [loop] go to Q3.

As an example of the Algorithm Q, consider common subexpression analysis. The pools are partitions of available expressions which are in the same class if they are known to have the same value. For example, the result of analyzing the statements

```
A := 2;
B := 3;
C := A + B;
```

could result in the pool

{A, 2 | B, 3 | C, A + B, B + A, 2 + 3, 3 + 2, 5}

where "|" is the delimiter for the different equivalence classes. The meet operator is defined as a simple intersection operator upon the partition elements, and the optimizing function builds new equivalence classes from existing equivalence classes, based upon operations at a particular node.

Upon termination of the Algorithm Q, the final optimization pool (P_f) for each node can be found by selecting the step in which each node was last processed. The current pool (P_c) at this step will be the final optimizing pool (P_f) which can be used in a subsequent pass through the block to perform the actual program transformations.

Consider the ALGOL program of Figure 1. The basic blocks are A, B, C and D with the entry node A and corresponding initial optimizing pool set to empty. Figure 2 gives the P_f for the program of Figure 1. Only a single pass of each block was required since there are no loops in the program flow graph. In general, the algorithm requires multiple passes on some blocks to produce the final optimizing pool. A complete and formal discussion of the global flow analysis algorithm by Kildall can be found in [28, 29, 30].

NODE	P_f
A	null
B	{x | y | x > y}
C	{a, b+c, c+b | y, x}
D	{a, h, b + c, c + d | y, x}

FIGURE 2

EXAMPLE OF FINAL POOL CONFIGURATION

II. IMPLEMENTATION OF ALGORITHM Q

A. CONCEPTUAL OVERVIEW

The original implementation of Algorithm Q was given by Kildall [30], and this project is basically an extension of Kildall's original efforts.

Implementation of the global flow analysis algorithm consists of three separate modules written in XPL [37]. The first module, the "Code Synthesis Filter Generator," or CSFG, creates tables which are used by the other two modules. A set of procedures known as the "Code Synthesis Filter Emitters," CSFE, constitute the second module which is used with the XPL skeletal compiler generator. The CSFE produces the intermediate language which is to be optimized. A particular compiler was implemented using the CSFE for a language called ALGOL-E, which is described elsewhere [32]. The third module implements the global flow analysis algorithm itself, and is called the "Code Synthesis Filter," or CSF.

Communication between the CSFE and CSF is through an intermediate language which represents a program in Polish form. The CSF reads the intermediate language from the compiler and flags optimization information, without actually performing any program transformations. Figure 3 outlines the flow of the present three module system.

The next two sections contain a description of the CSFG and CSFE as implemented by Kildall [30]. They are included here to ensure the completeness of the later CSF description and to simplify the discussion of the CSF module.

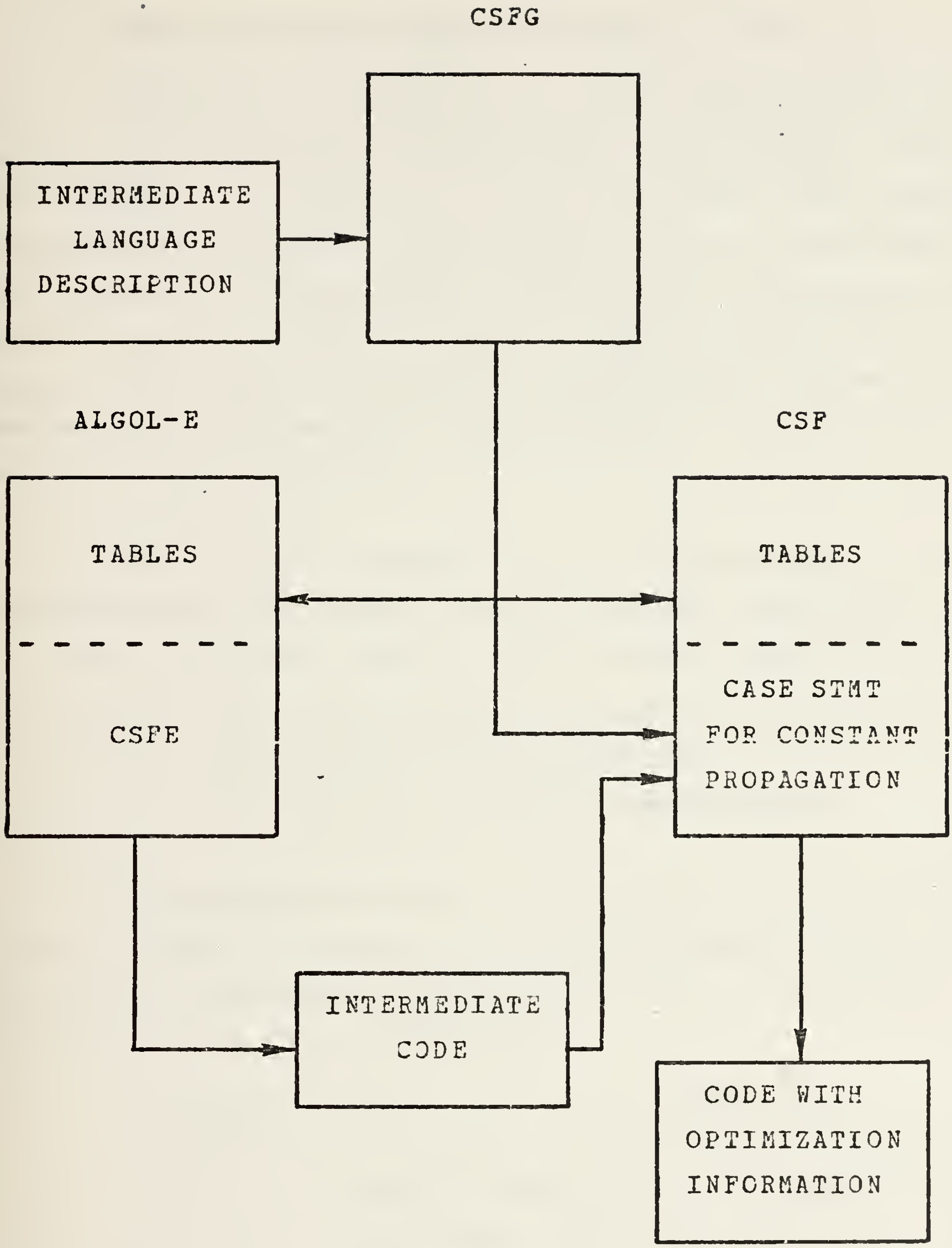

FIGURE 3

ORGANIZATION OF THE ALGORITHM Q IMPLEMENTATION

1. Code Synthesis Filter Generator (CSFG)

Primarily, the function of the CSFG is to provide a method for generating a customized intermediate language. The output of CSFG is a set of tables which define an intermediate language, along with an XPL case statement used for constant propagation. The CSF and CSFE are driven by these tables. The intermediate language takes the form of a string of Polish operators and operands, augumented with control flow information, which assumes a stack model for semantic interpretation.

The CSFG accepts an intermediate language specification determined by the grammar specified in TABLE I. There are three parts to the language specification:

1) <TYPE SPECIFICATION>
2) <OPCODE SPECIFICATION>
3) <SIMPLIFICATION SPECIFICATION>.

<TYPE SPECIFICATION> is primarily a list of variable types. Those followed by an asterisk denote data elements that occupy "regions" upon execution (i.e., an array). The following types are predeclared:

1) DEFAULT - type cannot be categorized,
2) LOC - location variable (i.e., address),
3) ENT - program entry point or branch operation destination point,
4) XIT - program branch point.

An XIT (exit) is the operand of a branch instruction while the ENT (entry) type variable is placed in the Polish sequence at the destination of a branch.

<OPCODE SPECIFICATION> gives the descriptive characteristics for each operator allowed in the Polish string. The following characteristics are generated for each OPCODE:

1) operator name
2) special attributes of the operator
3) number of operands necessary to perform the operation
4) type of each operand
5) the number of operands resulting from an operator
6) type of the resulting operands of an operator.

Special operators (REFER, PASS and TOGGLE) are predefined and require a LOC type argument with no resulting operands. These operators are used for special purposes discussed in Section II.A.2.

Special operator attributes aid in the optimization process by indicating operators which affect program flow or which have special effects on the execution stack. Specifically, these attributes are:

1) <u>LOAD</u> - results in a fetch from memory. The operand must be of type LOC (i.e., an address from which to fetch).

2) <u>STORE</u> - results in a store of the second operand into the address specified by the first operand. The first operand must be type LOC. The stack is cleared of both operands upon completion.

TABLE I

GRAMMAR FOR CSF GENERATION DESCRIPTION LANGUAGE

```
<CSF SPEC>  ::=  <TYPE SPEC> <OPCODE SPEC>
              |  <TYPE SPEC> <OPCODE SPEC>
              |  <SIMP SPEC>

<TYPE SPEC>  ::=  <TYPE HEAD> <TYPE ELEMENT> ;

<TYPE HEAD>  ::=  <TYPES>
              |  <TYPE HEAD> <TYPE ELEMENT> ,

<TYPES>  ::=  TYPES

<TYPE ELEMENT>  ::=  <IDENTIFIER>
                    <IDENTIFIER>  *

<OPCODE SPEC>  ::=  <OPCODE HEAD> <OPCODE RULE> ;

<OPCODE HEAD>  ::=  <OPCODES>
                 |  <OPCODE HEAD> <OPCODE RULE>

<OPCODES>  ::=  OPCODES

<OPCODE RULE>  ::=  <OPCODE FORMAT>
                 |  <OPCODE FORMAT> <TYPE>
                 |  <OPCODE FORMAT>  *

<OPCODE FORMAT>  ::=  <OPCODE PARMS> = >

<OPCODE PARMS>  ::=  <OPCODE PARM HEAD>
                  |  <OPCODE PARMS> <TYPE>
                  |  <OPCODE PARMS>  *

<OPCODE PARM HEAD>  ::=  <IDENTIFIER>
                      |  <IDENTIFIER>
                      |  (<OPCODE TYPE>)
```

TABLE I (CON'T)

```
<OPCODE TYPE>  ::=  LOAD
                 |  STORE
                 |  STORET
                 |  DELETE
                 |  DUPLIC
                 |  CONV
                 |  COMM
                 |  EXCH
                 |  COND
                 |  UCOND
                 |  GOSUB
                 |  RETSUB

<SIMP SPEC>  ::=  <SIMP HEAD> <SIMP RULE> ;

<SIMP HEAD>  ::=  <SIMPLIFICATIONS>
               |  <SIMP HEAD> <SIMP RULE> ,

<SIMPLIFICATIONS>  ::=  SIMPLIFICATIONS

<SIMP RULE>  ::=  <SIMP FORMAT> <IDENTIFIER>
               |  <SIMP FORMAT> <STRING>

<SIMP FORMAT>  ::=  <SIMP LEFT PART> = >

<SIMP LEFT PART>  ::=  <SIMP RULE HEAD> <IDENTIFIER>
                    |  <SIMP RULE HEAD> <STRING>
                    |  <SIMP RULE HEAD>  *

<SIMP RULE HEAD>  ::=  <OPCODE>
                    |  <SIMP RULE HEAD> <IDENTIFIER>
                    |  <SIMP RULE HEAD> <STRING>
                    |  <SIMP RULE HEAD>  *
```

3) STORET - the same as STORE; however, the second operand is retained on the stack for future use.

4) DELETE - removes the top entry from the stack (i.e., normally an operand).

5) DUPLIC - duplicates the top entry on the stack.

6) CONV - type conversion operators.

7) COMM - operator is communitive in all its operands.

8) EXCH - exchanges top two entries on the stack.

9) COND - conditional branch operator.

10) UCOND - unconditional branch operator.

11) GOSUB - branch to a procedure.

12) RETSUB - return branch from a procedure to the calling point.

<SIMPLIFICATION SPECIFICATION> defines acceptable transformations which may be performed on the operators. Each transformation is of the form

1) operator
2) operands
3) result operands.

For example:

ADD "0" A => A

indicates that the ADD operator would result in the operand A if one operand is zero, where A represents any arbitrary expression.

TABLE II outlines the particular intermediate language description used for the ALGOL-E compiler. TABLE III provides the complete output from the CSFG for this description.

2. Code Synthesis Filter Emitters (CSFE)

The CSFE is a set of XPL procedures that is included as an integral part of the XPL skeletal compiler for any particular compiler generation. The CSFE uses the tables produced by the CSFG to generate the intermediate language which is compatible with the optimizer, but tailored to the specific source language and compiler being developed.

As discussed above, the intermediate language is a Polish sequence consisting of a string of operators, operands, and branch point references. A table of referenced constants is also included in the Polish sequence. Figure 4 shows the format of the CSFE output. Individual fields are:

1) OFFSET(8b) - used for diagnostic reporting.

2) OPCODE/TYPE(8b) - designates the opcodes if the entry is an operator, or designates the type of operand if the entry is an operand.

3) C(1b) - if set, indicates a reference to the constant tables; if not, reference is to an address.

4) ADDR(15b) - address field of operands in the intermediate language.

TABLE II

A SAMPLE CSF GENERATOR SPECIFICATION FOR ALGOL-E

```
/* $OUTPUT PRINTED TABLES */
/* $PUNCH TABLES */
TYPES INT,REAL,BOOL,IARRAY*,RARRAY*;
OPCODES
TRU (CONV) REAL => INT                  /* TRUNCATE TAKES */
                                        /* REAL TO INTEGER */
RND (CONV) REAL => INT                  /* RND IS THE */
                                        /* ROUND OPERATOR */
FLT (CONV) INT => REAL,                 /* FLT IS THE */
                                        /* FLOAT FUNCTION */
ADD (COMM) INT INT => INT,              /* INTEGER ADDITION */
                                        /* (COMMUTATIVE) */
RADD (COMM) REAL REAL => REAL,          /* REAL ADDITION */
SUB INT INT => INT,                     /* INTEGER */
                                        /* SUBTRACTION */
RSUB REAL REAL => REAL,                 /* REAL SUBTRACTION */
MUL (COMM) INT INT => INT,              /* INTEGER */
                                        /* MULTIPLICATION */
RMUL (COMM) REAL REAL => REAL,          /* REAL SUBTRACTION */
DIV INT INT => INT,                     /* INTEGER DIVISION  */
RDIV REAL REAL => REAL,                 /* REAL DIVISION  */
EXP INT INT => INT,                     /* INTEGER */
                                        /* EXPONENTIATION */
RIXP REAL INT => REAL,                  /* REAL TO INTEGER  */
                                        /* EXPONENT */
IRXP INT REAL => REAL,                  /* INTEGER TO REAL  */
                                        /* EXPONENT */
RRXP REAL REAL => REAL,                 /* REAL TO REAL  */
                                        /* EXPONENT */
```

TABLE II (CON'T)

```
LSS * * => BOOL,                        /* LESS THAN TEST */
LEQ * * => BOOL,                        /* LESS THAN OR */
                                        /* EQUAL TEST */
EQL (COMM) * * => BOOL,                 /* EQUAL TO TEST */
NEQ * * => BOOL,                        /* NOT EQUAL TO TEST */
GEQ * * => BOOL,                        /* GREATER THAN OR */
                                        /* EQUAL TO */
GTR * * => BOOL,                        /* GREATER THAN TEST */
INEG INT => INT,                        /* INTEGER  NEGATION */
RNEG REAL => REAL,                      /* REAL NEGATION */
NOT BOOL => BOOL,                       /* BOOLEAN NOT */
AND (COMM) BOOL BOOL => BOOL,           /* BOOLEAN AND */
BOR (COMM) BOOL BOOL => BOOL,           /* BOOLEAN OR */
LOD (LOAD) LOC => * ,                   /* LOAD FROM ADDRESS */
STO (STORET) LOC * => *,                /* STORE AND RETAIN */
                                        /* VALUE */
STD (STORE) LOC * => ,                  /* STORE AND REMOVE */
                                        /* VALUE */
DEL (DELETE) * => ,                     /* DELETE TOP */
                                        /* ELEMENT */
DUP (DUPLIC)  => *,                     /* DUPLICATE */
                                        /* ELEMENT */
XCH (EXCH) => ,                         /* EXCHANGE LAST */
                                        /* ELEMENTS */
HLT => ,                                /* PROGRAM HALT */
BRS (UCOND) XIT => ,                    /* UNCONDITIONAL */
                                        /* BRANCH */
BSC (COND) BOOL XIT => ,                /* CONDITIONAL */
                                        /* BRANCH */
NOP => ,                                /* NO OPERATION */
```

TABLE II (CON'T)

```
PRO (GOSUB) XIT => ,                /* PROCEDURE   */
                                    /* TRANSFER */
RTN (RETSUB) => ,                   /* RETURN FROM */
                                    /* PROCEDURE */
GET INT => LOC,                     /* GET STORAGE */
RET LOC => ,                        /* RETURN STORAGE */
RRD => REAL,                        /* REAL READ */
IRD => INT,                         /* INTEGER READ */
BRD => BOOL,                        /* BOOLEAN READ */
WRVx* => ,                          /* WRITE VARIABLE */
DMP => ,                            /* DUMP BUFFER */
TAB INT => ,                        /* TABULATE */
SUP * => ;
SIMPLIFICATIONS
ADD "0" A => A,                     /* 0+A=A+0=A */
RADD "0" A => A,
MUL "0" A => "0",                   /* 0*A=A*0=0 */
RMUL "0" A => "0",
MUL "1" A => A,                     /* 1*A=A*1=A */
RMUL "1" A => A,
SUB A A => "0",                     /* A-A= */
RSUB A A => "0",
DIV "0" A => "0" ,                  /* 0/A = 0 */
RDIV "0" A => "0" ,                 /* 0/A = 0 */
EXP A "0" => "1" ,                  /* A**0 = 1 */
RIXP A "0" => "1" ,
IRXP A "0" => "1" ,
RRXP A "0" => "1" ,
EXP A "1" => A ,                    /* A ** 1 = A */
RIXP A "1" => A,                    /* A ** 1 = A */
IRXP A "1" => A,
RRXP A "1" => A,
```

TABLE III

CSFG OUTPUT FOR ALGOL-E

```
DECLARE(NULL, DEFAULT, LOC, ENT, XIT, INT, REAL, BOOL, IARRA
Y, RARRAY) BIT(8);
DECLARE NTYPES LITERALLY'9';
DECLARE(REFER, TOGGLE, PASS, TRU, RND, FLT, ADD, RADD, SUB,
RSUB, MUL, RMUL, DIV, RDIV, EXP, RIXP, IRXP, RRXP, LSS, LEQ,
 EQL, NEQ, GEQ, GTR, INEG, RNEG, NTO, AND, BOR, LOD, STO, ST
D, DEL, DUP, XCH, HLT, BRS, BSC, NOP, PRO, RTN, GET, RET, RR
D, IRD, BRD, WRV, DMP, TAB, SUP) BIT(8);
DECLARE NOPCODES LITERALLY'50';
DECLARE OPCODES CHARACTER INITIAL(
'NULLDEFAULTLOCENTXITINTREALBOOLIARRAYRARRAYREFERTOGGLEPASST
RURNDFLTADDRADDSUBRSUBMULRMULDIVRDIVEXPRIXPIRXPXPRRXPLSSLEQE
QLNEQGEQGTRINEGRNEGNOTANDBORLODSTOSTDDELDUPXCHHLTBRSBSCNOPPR
ORTNGETRETRRDIRDBRDWRVDMPTABSUP');
DECLARE OPRANGE(60) BIT(8) INITIAL(0, 4, 11, 14, 17, 20, 23,
27,  31, 37, 43, 48, 54, 58, 61, 64, 67, 70, 74, 77, 81, 84,
88, 91, 95, 98, 102, 106, 110, 113, 116, 119, 122, 125, 128,
132,  136, 139, 142, 145, 148, 151, 154, 157, 160, 163, 166,
169, 172, 175, 178, 181, 184, 187, 190, 193, 196, 199,  202,
205, 205);
DECLARE OP_TYPE(59) BIT(8) INITIAL(0, 0, 0, 0, 0, 0,  0,  0,
1,  1, 0, 0, 0, 7, 7, 7, 8, 8, 0, 0, 8, 8, 0, 0, 0, 0, 0, 0,
0, 0, 8, 0, 0, 0, 0, 0, 0, 8, 8, 2, 4, 3, 5, 6,  9,  0,  11,
10, 0, 12, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0);
DECLARE OP_DEGL(59) BIT(8) INITIAL(0, 0, 0, 0, 0, 0,  0,  0,
0,  0, 0, 0, 0, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 1, 1, 1, 2, 2, 1, 2, 2, 1, 0, 0, 0, 1,  2,
0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1);
```

TABLE III (CON'T)

```
DECLARE OP_DEGR(59) BIT(8) INITIAL(0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0);
DECLARE OP_INDEX(60) BIT(8) INITIAL(0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 3, 5, 7, 10, 13, 16, 19, 22, 25, 28, 31,
34, 37, 40, 43, 46, 49, 52, 55, 58, 61, 63, 65, 67, 70, 73,
75, 78, 80, 81, 82, 82, 82, 83, 85, 85, 86, 86, 88, 89, 90,
91, 92, 93, 93, 94, 95);
DECLARE  OP_INFO(170) BIT(16) INITIAL(0,  6, 5, 6, 5, 5, 6,
5, 5, 5, 6, 6, 6, 5, 5, 5, 6, 6, 6, 5, 5, 5, 6, 6, 6, 5, 5,
5, 6, 6, 6, 5, 5, 5, 6, 5, 6, 5, 6, 6, 6, 6, 6, 1, 1, 7, 1,
1, 7, 1, 1, 7, 1, 1, 7, 1, 18 7, 1, 1, 7, 5, 5, 6, 6, 7, 7,
7, 7, 7, 7, 7, 7, 2, 1, 2, 1, 1, 2, 1, 1, 1, 4, 7, 4, 4, 5,
2, 2, 6, 5, 7, 1, 5, 1, 0, 3, 2, 2, 0, 3, 2, 2, 111, 3, 2,
3, 115, 3, 2, 3, 0, 5, 2, 2, 0, 5, 2, 2, 0, 2, 2, 3, 0, 2,
2, 3, 0, 3, 2, 3, 0, 3, 2, 3, 151, 2, 3, 5, 155, 2, 3, 5,
159, 2, 3, 5, 163, 2, 3, 5, 0, 2, 5, 2, 0, 2, 5, 2, 0, 2, 5,
2, 167, 2, 5, 2, 0, 2, 5, 2);
DECLARE  SIMP_INDEX(59) BIT(8) INITIAL(0,  0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 95, 99, 119, 123, 103, 107, 127,
131 135, 139, 143, 147, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 08 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
DECLARE OP_STR(2) CHARACTER INITIAL('', '0', '1');
**** **** **** **** **** ****
DECLARE(NULL, DEFAULT, LOC, ENT, XIT, INT, REAL, BOOL, IARRA
Y, RARRAY)  BIT(8);
DECLARE NTYPES LITERALLY'9';
```

TABLE III (CON'T)

```
DECLARE(REFER, TOGGLE, PASS, TRU, RND, FLT, ADD, RADD, SUB,
RSUB, MUL, RMUL, DIV, RDIV, EXP, RIXP, IRXP, RRXP, LSS, LEQ,
 EQL, NEQ, GEQ, GTR, INEG, RNEG, NOT, AND, BOR, LOD, STO, ST
D, DEL, DUP, XCH, HLT, BRS, BSC, NOP, PRO, RTN, GET, RET, RR
D, IRD, BRD, WRV, DMP, TAB, SUP) BIT(8);
DECLARE NOPCODES LITERALLY'50';
DECLARE OPCODES CHARACTER INITIAL(
'NULLDEFAULTLOCENTXITINTREALBOOLIARRAYRARRAYREFERTOGGLEPASST
RURNDFLTADDRADDSUBRSUBMULRMULDIVRDIVEXPRIXPIRXPRRXPLSSLEQEQL
NEQGEQGTRINEGRNEGNOTANDBORLODSTOSTDDELDUPXCHHLTBRSBSCNOPPROR
TNGETRETRRDIRDBRDWRVDMPTABSUP');
DECLARE  OPRANGE(60) BIT(8) INITIAL(0,  4,  11, 14,  17, 20,
23, 27, 31, 37, 43, 48, 54, 58, 61, 64, 67, 70, 74, 77,  81,
84,  88, 91, 95, 98, 102, 106, 110, 113, 116, 119, 122, 125,
128, 132, 136, 139, 142, 145, 148, 151, 154, 157, 160,  163,
166,  169, 172, 175, 178, 181, 184, 187, 190, 193, 196, 199,
202, 205, 208);
DECLARE  OP_TYPE(59) BIT(8) INITIAL(0,  0, 0, 0, 0, 0, 0, 0,
1, 1, 0, 0, 0, 7, 7, 7, 8, 8, 0, 0, 8, 8, 0, 0, 0, 0, 0,  0,
0,  0,  8,  0, 0, 0, 0, 0, 0, 8, 8, 2, 4, 3, 5, 6, 9, 0, 11,
10, 0, 12, 13, 08 0, 0, 0, 0, 0, 0, 0, 0);
DECLARE  OP_DEGL(59) BIT(8) INITIAL(0,  0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,  2,
2,  2, 2, 2, 2, 2, 1, 1, 1, 2, 2, 1, 2, 2, 1, 0, 0, 0, 1, 2,
0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1);
DECLARE  OP_DEGR(59) BIT(8) INITIAL(0,  0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  1,
1,  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0);
```

```
DECLARE  OP_INDEX(60) BIT(8) INITIAL(0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 3, 5, 7, 10, 13, 16, 19, 22, 25,  28,  31,
34,  37, 40, 43, 46, 49, 52, 55, 58, 61, 63, 65, 67, 70, 73,
75, 78, 80, 81, 82, 82, 82, 83, 85, 85, 86, 86, 88, 89,  90,
91, 92, 93, 94, 95);
DECLARE OP_INFO(170) BIT(16) INITIAL(0,  6, 5, 6, 5,  5,  6,
5,  5, 5, 6, 6, 6, 5, 5, 5, 6, 6, 6, 5, 5, 5, 6, 6, 6, 5, 5,
5, 6, 6, 6, 5, 5, 5, 6, 5, 6, 5, 6, 6, 6, 6, 6, 1, 1, 7,  1,
1,  7, 1, 1, 7, 1, 1, 7, 1, 1, 7, 1, 1, 7, 5, 5, 6, 6, 7, 7,
7, 7, 7, 7, 7, 7, 2, 1, 2, 1, 1, 2, 1, 1, 1, 4, 7, 4, 4,  5,
2,  2,  6, 5, 7, 1, 5, 1, 0, 3, 2, 2, 0, 3, 2, 2, 111, 3, 2,
3, 115, 3, 2, 3, 0, 5, 2, 2, 0, 5, 2, 2, 0, 2, 2, 3,  0,  2,
2,  3,  0,  3, 2, 3, 0, 3, 2, 3, 151, 2, 3, 5, 155, 2, 3, 5,
159, 2, 3, 5, 163, 2, 3, 5, 0, 2, 5, 2, 0, 2, 5, 2, 0, 2, 5,
2,  167, 2, 5, 2, 0, 2, 5, 2);
DECLARE SIMP_INDEX(59) BIT(8) INITIAL(0,  0, 0, 0, 0, 0,  0,
0,  0,  0, 0, 0, 0, 0, 0, 0, 95, 99, 119, 123, 103, 107,127,
131, 135, 139, 143, 147, 0, 0, 0, 08 08 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
DECLARE OP_STR(2) CHARACTER INITIAL('', '0', '1');
   /* CASE 0: NULL */
        ;
   /* CASE 1: DEFAULT =>  (TYPE) */
        ;
   /* CASE 2: LOC =>  (TYPE) */
        ;
   /* CASE 3: ENT =>  (TYPE) */
        ;
   /* CASE 4: XIT =>  (TYPE) */
        ;
   /* CASE 5: INT =>  (TYPE) */
        ;
```

TABLE III (CON'T)

```
.

/* CASE 6: REAL =>  (TYPE) */
      ;
/* CASE 7: BOOL =>  (TYPE) */
      ;
/* CASE 8: IARRAY =>  (TYPE) */
      ;
/* CASE 9: RARRAY =>  (TYPE) */
      ;
/* CASE 9: RARRAY =>  (TYPE) */
      ;
/* CASE 10: REFER =>  (OPCODE) */
      ;
/* CASE 11: TOGGLE =>  (OPCODE) */
      ;
/* CASE 12: PASS =>  (OPCODE) */
      ;
/* CASE 13: TRU REAL => INT  (OPCODE) */
      ;
/* CASE 14: RND REAL => INT  (OPCODE) */
      ;
/* CASE 15: FLT INT => REAL  (OPCODE) */
      ;
/* CASE 16: ADD INT INT => INT  (OPCODE   */
      ;
/* CASE 17: RADD REAL REAL => REAL  (OPCODE) */
      ;
/* CASE 18: SUB INT INT => INT  (OPCODE)  */
      ;
/* CASE 19: RSUB REAL REAL => REAL  (OPCODE) */
      ;
/* CASE 20: MUL INT INT => INT  (OPCODE)  */
      ;
/* CASE 21: RMUL REAL REAL => REAL  (OPCODE) */
```

TABLE III (CON'T)

```
      ;
/* CASE 22: DIV INT INT => INT  (OPCODE)   */
      ;
/* CASE 23: RDIV REAL REAL => REAL  (OPCODE) */
      ;
/* CASE 24: EXP INT INT => INT  (OPCODE)   */
      ;
/* CASE 25: RIXP REAL INT => REAL  (OPCODE) */
      ;
/* CASE 26: IRXP INT REAL => REAL  (OPCODE) */
      ;
/* CASE 27: RRXP REAL REAL => REAL  (OPCODE) */
      ;
/* CASE 28: LSS DEFAULT DEFAULT => BOOL  (OPCODE) */
      ;
/* CASE 29: LEQ DEFAULT DEFAULT => BOOL  (OPCODE) */
      ;
/* CASE 30: EQL DEFAULT DEFAULT => BOOL  (OPCODE) */
      ;
/* CASE 31: NEQ DEFAULT DEFAULT => BOOL  (OPCODE) */
      ;
/* CASE 32: GEQ DEFAULT DEFAULT => BOOL  (OPCODE) */
      ;
/* CASE 33: GTR DEFAULT DEFAULT => BOOL  (OPCODE) */
      ;
/* CASE 34: INEG INT => INT  (OPCODE) */
      ;
/* CASE 35: RNEG REAL => REAL  (OPCODE)   */
      ;
/* CASE 36: NOT BOOL => BOOL  (OPCODE)   */
      ;
/* CASE 37: AND BOOL BOOL => BOOL  (OPCODE */
```

TABLE III (CON'T)

```
      ;
/* CASE 38: BOR BOOL BOOL => BOOL  (OPCODE */
      ;
/* CASE 39: LOD LOC => DEFAULT  (OPCODE   */
      ;
/* CASE 40: STO LOC DEFAULT => DEFAULT  (OPCODE) */
      ;
/* CASE 41: STD LOC DEFAULT => (OPCODE) */
      ;
/* CASE 42: DEL DEFAULT =>  (OPCODE) */
      ;
/* CASE 43: DUP => DEFAULT  (OPCODE) */
      ;
/* CASE 44: XCH =>  (OPCODE) */
      ;
/* CASE 45: HLT =>  (OPCODE) */
      ;
/* CASE 46: BRS XIT =>  (OPCODE) */
      ;
/* CASE 47: BSC BOOL XIT =>  (OPCODE) */
      ;
/* CASE 48: NOP =>  (OPCODR) */
      ;
/* CASE 49: PRO XIT =>  (OPCODE) */
      ;
/* CASE 50: RTN =>  (OPCODE) */
      ;
/* CASE 51: GET INT => LOC  (OPCODE) */
      ;
/* CASE 52: RET LOC =>  (OPCODE) */
      ;
/* CASE 53: RRD => REAL  (OPCODE) */
```

```
        ;
/* CASE 54: IRD => INT  (OPCODE) */
        ;
/* CASE 55: BRD => BOOL  (OPCODE) */
        ;
/* CASE 56: WRV DEFAULT =>  (OPCODE) */
        ;
/* CASE 57: DMP =>  (OPCODE) */
        ;
/* CASE 58: TAB INT =>  (OPCODE) */
        ;
/* CASE 59: SUP DEFAULT =>  (OPCODE) */
        ;
```

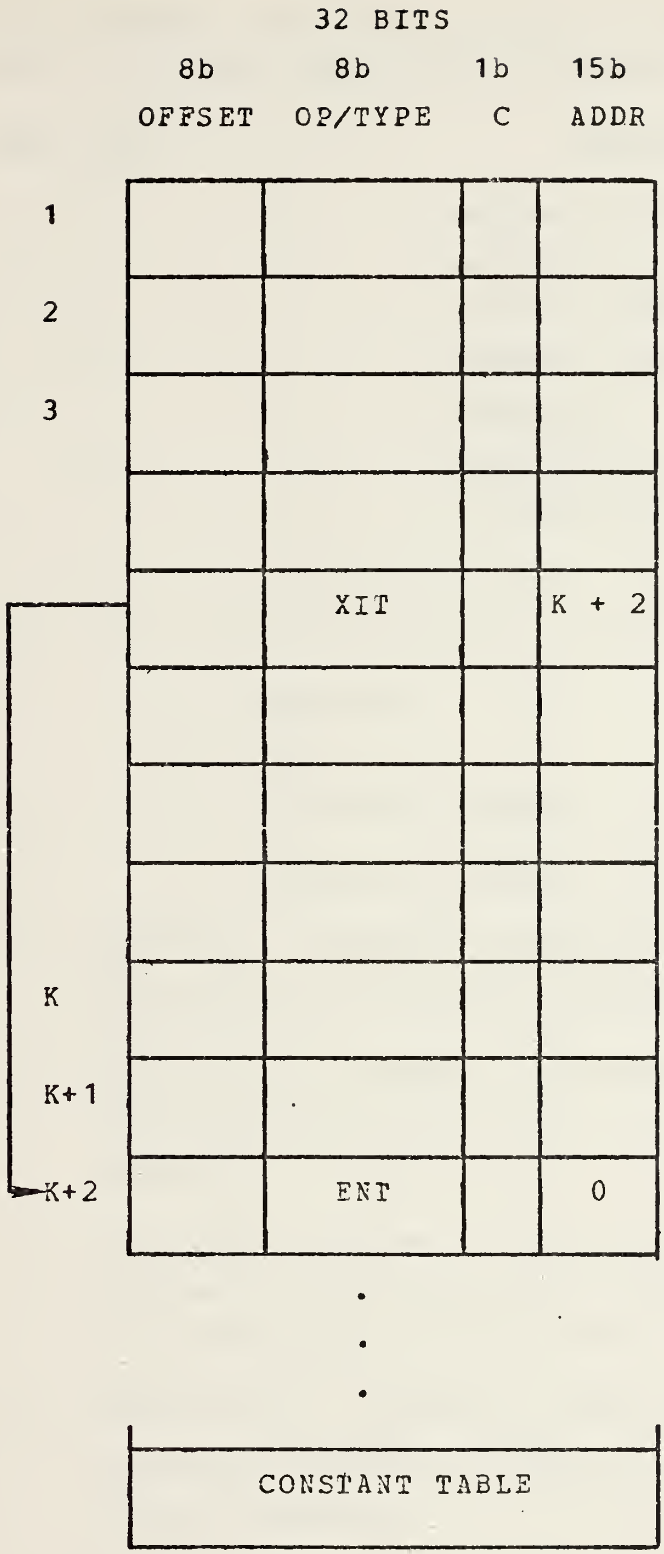

FIGURE 4

INTERMEDIATE LANGUAGE OUTPUT FORMAT FROM CSFE

The CONSTANT TABLE begins with a count of the number of constants. An entry then follows for each constant as:

BYTE (8b)	CONTENT
0	number of characters for the constant
1	type of constant
2-3	address assigned to the constant
4+i	character string of length i representing the constant.

The basic CSFE procedures are as listed with calling values of:

o - operator

t - type

s - character string

x - integer constant $0 \le x \le 2^{14}$.

1) EMITO(o) - places a reference to an operator in the Polish sequence.

2) EMITC(s,t) - places a reference to the constant string s of type t into the Polish sequence.

3) EMITA(x) - places a reference to address x into the Polish sequence. Type LOC is associated with the address.

4) EMITAC(s,t) - places a reference to the constant string s of type t into the Polish sequence.

5) EMITI(t) - places a reference to an operand of type t with an indeterminate value into the Polish sequence.

Additionally, the CSFE handles "branch point resolution." Branch points are referenced symbolically in the Polish sequence during compilation. A particular point in the sequence can be given a symbolic name for future reference. Whenever a symbolic location is defined, an ENT is inserted. Furthermore, references can be made to locations which have not as yet been compiled, called an "unresolved reference." When an unresolved reference occurs, the CSFE "resolves" the reference by inserting an address in the ADDR field of all XIT operands that previously referenced the symbol.

The CSFE procedures for label processing are:

1) EMITE(s) - ENT type variable which resolves the top-most stacked reference to s.

2) EMITB(s) - backward reference to a label s which has already occurred.

3) EMITF(s) - forward branch point which has not occurred.

4) EMITPE(s) - ENT type variable that does not resolve any references to label s, called a "push down entry."

5) EMITPF(s) - forward reference to label s which "pushes" all previous references to label s. An EMITE(s) will resolve this reference but not the references which were "pushed down."

6) SAVELAB - saves the current state of the label stack. Used to process inner loops such as embedded for-loops.

7) RESTORELAB - restores the label stack to the configuration present upon the last call to SAVELAB.

Additional utility procedures are also present as follows:

1) EMITR(x) - places a REFER operator into the Polish sequence. The ADDR field is set to x. This is normally called at each card boundary in the source program with a line number of x; therefore, if an error condition is detected, the Polish sequence is scanned backwards to the last REFER. The ADDR field along with the OFFSET field at the point of error are combined to pinpoint the error in the original source program.

2) EMITP(x) - places a PASS operator with an ADDR field of x into the Polish sequence. The PASS operator is ignored by the CSF. Thus, the PASS operator can be used to pass information to subsequent modules.

3) EMITT(x) - places a TOG operator with an ADDR field of x into the Polish sequence. The x value is a bit word representation of toggle flags that can be set on the compile pass for later detection by CSF. Compilation toggles are enabled or disabled in the ALGOL-E implementation by:

a) $CODE - results in the Polish sequence being interlisted with the source program.

b) $TABLES - causes CSF to give a full trace by table dumps of the optimization process.

c) $EXEC - causes a partial trace of the CSF analysis.

d) $TRACE - causes CSF to give a partial trace of the optimization process.

During the interlisting of the Polish sequence, (i.e., $CODE is selected), each line of the code trace will hold

1) OFFSET field - not used

2) OPCODE/TYPE - operator or operand type

3) C field - constant indicator

4) ADDR field - address

5) 32-bit word in hexadecimal format.

A "+" before a line indicates the resolution of a forward reference while a "++" indicates that the entry point at that line number has caused the resolution of a forward reference or has been referenced again as a backward reference. The ADDR field of each ENT variable is incremented on each reference from an XIT variable. Therefore, it contains the number of branches into the location of ENT. TABLE IV is an example of the code interlisting option.

The "constant table" is appended to the end of the Polish sequence. This table provides a character string representation of each constant encountered in the source program. The entire code file (i.e., Polish sequence and constant table) is passed to the CSF at the end of compilation.

B. OPTIMIZING FUNCTIONS INCORPORATED IN THE CSF

Before giving the details of the CSF implementation, it is necessary to describe the optimizing function and meet operator currently used. Information for three forms of optimization is collected:

1) Constant Propagation
2) Simplifying Formal Identities
3) Common Subexpression Elimination.

1. Constant Propagation

The pool associated with each basic block is a partition of the program expressions. Constant propagation requires that a constant indicator be associated with every equivalent class. For example:

```
A := 2;
B := 3;
C := A + B;
```

could be rewritten as

```
C := 5;
```

since A and B are both known constants. Therefore, the pool associated with each node is a partition with a constant indicator for each equivalence class. The meet operator combines the constant indicators.

TABLE IV

ALGOL-E CODE INTERLISTED PROGRAM

CARD NUMBER
BLOCK NUMBER
POLISH ELEMENT COUNTER
HEXADECIMAL FORM

```
CARD |BL| SYL |
  1  | 0|   0 |COMMENT      TEST PROGRAM # 10        ;
$EXECUTE
$CODE     CODE
$TABLES   LOCATION  OFFSET  OPCODE/TYPE C-FIELD  ADDRESS  HEXADECIMAL FORM
                2      0      |TOGGLE     |  258   | 00080102
  5  | 0|   2 |BEGIN LOCAL A,B,C,D,E;
  6  | 1|   2 |    C := 2;
                3      0      |LOC        |  3     | 00020003
                4      0      |INT        |  1     | 00050001
                5      0      |STO        |  0     | 00280000
                6      0      |DEL        |  0     | 002A0000
  7  | 1|   6 |    D := 3;
                7      0      |LOC        |  4     | 00020004
                8      0      |INT        |  2     | 00050002
                9      0      |STO        |  0     | 00280000
               10      0      |DEL        |  0     | 002A0000
  8  | 1|  10 |    READ(A,B);
               11      0      |LOC        |  1     | 00020001
               12      0      |INT        |  0     | 00050000
               13      0      |STD        |  0     | 00290000
               14      0      |LOC        |  2     | 00020002
               15      0      |INT        |  0     | 00050000
               16      0      |STD        |  0     | 00290000
  9  | 1|  16 |    WHILE A GTR B DO
               17      0      |ENT        |  0     | 00030000
               18      0      |LOC        |  1     | 00020001
               19      0      |LOD        |  0     | 00270000
               20      0      |LOC        |  2     | 00020002
               21      0      |LOD        |  0     | 00270000
               22      0      |GTR        |  0     | 00210000
               23      0      |XIT        |  0     | 00040000
               24      0      |XIT        |  26    | 0004001A
               25      0      |BSC        |  0     | 002F0000
               26      0      |ENT        |  1     | 00030001
 10  | 1|  26 |        B := C + D;
               27      0      |LOC        |  2     | 00020002
               28      0      |LOC        |  3     | 00020003
               29      0      |LOD        |  0     | 00270000
               30      0      |LOC        |  4     | 00020004
               31      0      |LOD        |  0     | 00270000
               32      0      |ADD        |  0     | 00100000
               33      0      |STO        |  0     | 00280000
               34      0      |DEL        |  0     | 002A0000
               35      0      |XIT        |  17    | 00040011
              +17      0      |ENT        |  1     | 00030001
               36      0      |BRS        |  0     | 002E0000
               37      0      |ENT        |  0     | 00030000
             + 23      0      |XIT        |  37    | 00040025
             ++37      0      |ENT        |  1     | 00030001
 11  | 1|  37 |    E := D + C;
               38      0      |LOC        |  5     | 00020005
               39      0      |LOC        |  4     | 00020004
               40      0      |LOD        |  0     | 00270000
               41      0      |LOC        |  3     | 00020003
               42      0      |LOD        |  0     | 00270000
               43      0      |ADD        |  0     | 00100000
               44      0      |STO        |  0     | 00280000
               45      0      |DEL        |  0     | 002A0000
 12  | 1|  45 |END
 13  | 1|  45 |EOF
CODE FILE COPIED (45 WORDS)
CONSTANT TABLE COPIED (4 WORDS)
2 RECORDS WRITTEN INTO FILE 1
END OF COMPILATION FEBRUARY 9, 1975.  CLOCK TIME = 18:2:58.02.

13 CARDS WERE READ.
NO ERRORS WERE DETECTED.
```

COMPILER CONTROL CARDS

ALGOL-E SOURCE

```
BEGIN LOCAL A,B,C,D,E;
      C := 2;
      D := 3;
      READ (A,B);
      WHILE A GTR B DO
          B := C + D;
      E := D + C;
      END
      EOF
```

BRANCH POINT RESOLUTION

2. Simplifying of Formal Identities

A list of the simplifying formal identities specified in the CSFG are available in tabular form in the CSF. During the analysis, the expressions are compared with this table and transformations are made if a match is found. For example, the expression

A := A + 0;

matches the form of the additive identity and could, therefore, be eliminated. Thus, no optimizing function or meet operator need be specified since these transformations are all local.

3. Common Subexpression Elimination

As described in Section I. C., the pools for common subexpression elimination consist of partitions of program expressions into groups that are known to have identical values. The meet operator is defined as the intersection on the equivalent classes.

The combination of the above three optimization forms are incorporated into the current CSF. This combination logically allows detection of a fourth optimization form, common subexpressions that are linked to a previous constant propagation. The meet operator is the intersection operator. The four types are detected using a hierarchy of:

1) Propagated Constants
2) Simplifying Formal Identities
3) Common Subexpressions that link to a Propagated Constant
4) Common Subexpressions.

C. CODE SYNTHESIS FILTER (CSF)

The following sections give the programming structures and methodology that are used in the implementation of the global flow analysis algorithm. The CSF module was written in the XPL language utilizing the XCOM control system [37].

1. Program Flow

The actual implementation of the global flow analysis algorithm is incorporated in this module. In order to analyze the input Polish sequence, a "meta-execution" is performed on the intermediate language. This process utilizes the "value number" concept of Cocke [10] which results in a more simplified data structure than would normally be expected. The "value number" will be referred to as a "class number," and an equivalence class will have one such class number assigned to it.

The CSF is separated into three logical parts:

1) Initialization
2) Control Flow Analyzer (CFA)
3) Basic Block Analyzer (BBA).

The initialization routines accept the intermediate code file from the CSFE and prepare all data structures for processing. It is here that a CONSTANT TABLE is built as produced by the ALGOL-E compiler.

The CFA controls the flow of processing from block to block, performs the meet operation and terminates the CSF.

The BBA is invoked to analyze each block (i.e., the application of the optimization function). It is here that meta-execution takes place and optimization information is detected. Figure 5 depicts the overall flow of the CSF.

The enclosed listing of CSF has been fully annotated in order to provide documentation of individual routines. A complete discussion of table structures and the more important procedures are given in the following sections.

2. Implementation Methodology

The implementation of the global flow analysis algorithm was done using structured programming concepts. All functions are isolated within procedures and each procedure is as autonomous as possible. A complete description of the program organization and output can be accomplished by examining four areas:

1) Control Structures
2) Meta-Execution
3) Meet Operation
4) Output Formats.

a. Control Structures

The program branching structure is followed by a procedure called "CONTROL-FLOW," the Control Flow Analyzer (CFA). The CFA repeatively cycles through the following steps:

1) select a block to be processed from the top of the list L.

2) call the Basic Block Analyzer (BBA) to process the block.

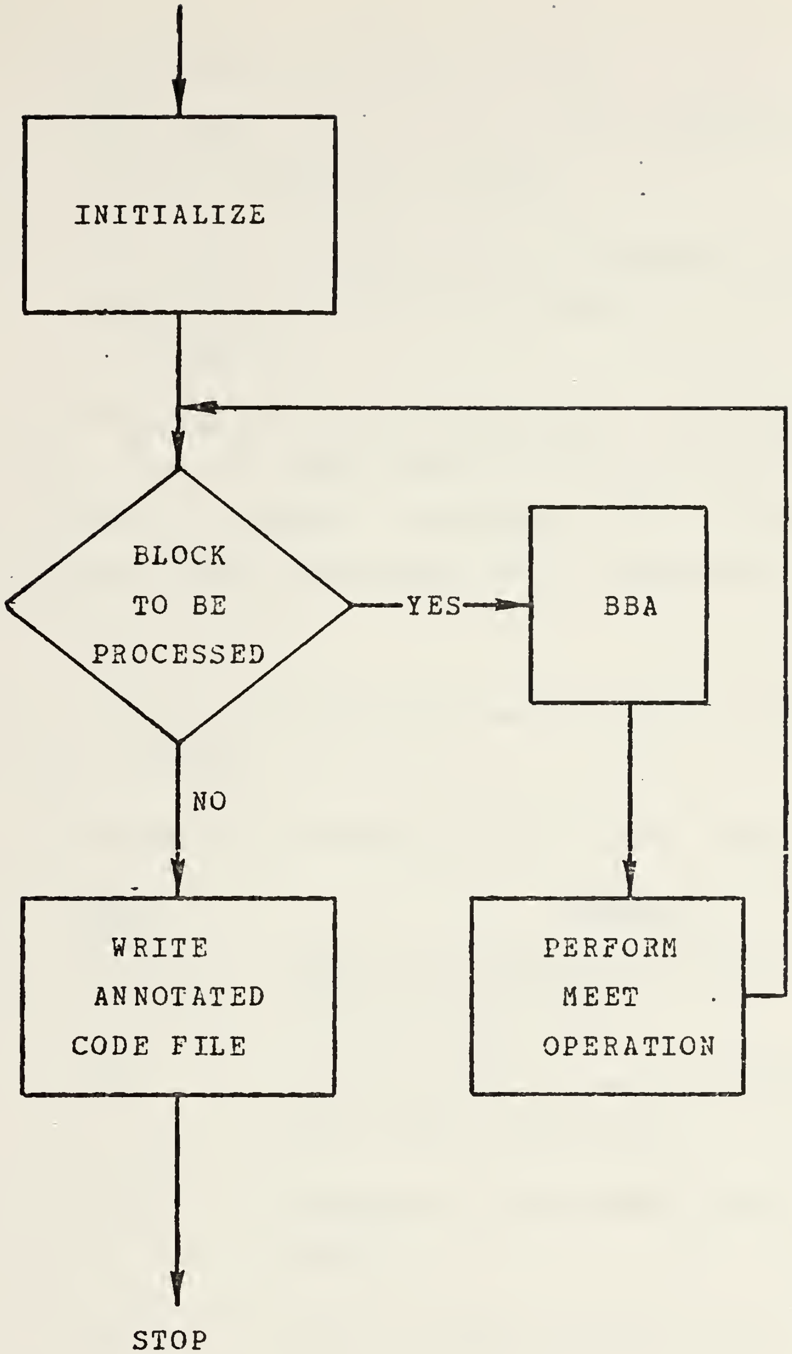

FIGURE 5

OVERALL FLOW OF CSF

3) use the P_o produced by the the BBA to perform the meet operation on all successor blocks.

4) order the blocks to be processed list according to the block selection rule in effect.

These steps are continued until no blocks are available to be processed in step 1. The CFA then calls a procedure, "OUTPUT_OPTIMIZATION," which writes an annotated intermediate language code file along with a synopsis of the optimization process.

The "blocks to be processed list," L, consists of two elements as follows:

1) CONTROL is a thirty-two (32) bit entry of:

BITS	CONTENT
0-15	first word address of the block in the code file,
16	set if initial pool has been built for this block,
17-30	immediate predecessor block number,
31	set if block is not to be processed (i.e., $P_c \leq P_i$ in step Q5 of the flow algorithm).

2) SAV-TOG is an eight (8) bit entry of:

BIT	
0	set if control flow trace of optimization process desired,
1	set if complete table trace of optimization process desired,
2	set if detected constant operands are to be propagated,
3	set if false branches are to be removed from detected constant decision branches,
4	set if output from CSF is desired in punched cards,
5-7	are not currently used.

The function of the two tables is to save blocks to be processed along with optimizer toggles for each block. If it is found that a block need not be processed (i.e., the test of $P_c \leq P_i$ is true after performing the meet operation in step Q5 of the algorithm) a single bit can be set rather than rearranging the table sequence. If an immediate successor block has never been encountered before, its P_c becomes the P_o from the immediate predecessor block in the procedure "BUILD_INPUT_POOL."

Blocks are entered on the top of the "to be processed list," L as they are encountered by the BBA (i.e., when a branch operator is encountered). After performing the meet operation on all successor blocks, the CFA calls

the procedure "ORDER_BLOCK_LIST," which arranges the "to be processed list," L, in a top down sequence for actual processing according to the specified block selection method. Presently, any one of four (4) block selection methods can be specified. The are:

1. Last-in-First-out (LIFO).

2. First-in-First-out (FIFO).

3. Select the P_i with the minimum number of equivalence classes called the min pool selection method, as discussed by Lukasczyk [34].

4. Select the P_i with the maximum number of equivalence classes called the max pool selection method. This was included for comparison against the min pool selection method.

In order to control block processing, a "block header" for each block is generated when it is first encountered. This header is updated each time the block is traversed; consequently, it provides a summary for the block at any point during the optimization process. These block headers are saved sequentially above the intermediate language code file. The block header format is:

WORD	CONTENT
0	Block # (15 bits) | previously processed indicator (1 bit) | STATES TABLE index (16 bits)
1	First word address in code file (16 bits) | last word address in code file (16 bits)

2	Number of transversals
3	Number of references
4	Number of predecessor blocks
.	referencing this block
.	(variable number of
.	entries).

Prior to passing a block's first word address to the BBA, the CFA places its P_i on the ADDRESS and EXPRESSION TABLES using information from the STATES TABLE. These tables are among the primary mechanisms for the meta-execution performed by the BBA and will be discussed in the next section. In this manner, the meta-execution tables are preset for any block thus allowing random processing of blocks.

The meet operation is performed by the CFA through a call to the procedure PERFORM_MEET_OP which applies the meet operator to a single immediate successor block of the block just processed. A true response from the PERFORM_MEET_OP procedure indicates that the immediate successor block is to be processed (i.e., the test of $P_c \leq P_i$ in step Q5 of Algorithm Q). As was mentioned previously, a single bit in the CONTROL TABLE entry for each successor block is then set or cleared to indicate future processing. The meet operation will be discussed in more detail in Section II.C.2.c.

b. Meta-Execution

The meta-execution is performed within the procedure BASIC_BLOCK which is the control routine for the BBA. The Polish sequence is symbolically evaluated using an

EXECUTION STACK. This process uses "class numbers" instead of the actual value of a variable since the actual value is not generally known at compile time. The EXECUTION STACK has four (4) elements:

1) EXVAL# - a sixteen bit entry which holds the class number.

2) EXTYPE - an eight bit entry which holds the type of an operand (e.g., BOOL or INT).

3) EXCON - a one bit indicator to identify an entry as a constant.

4) EXADD - a sixteen bit entry that points to the ADDRESS TABLE entry that corresponds to this EXECUTION STACK entry.

Three other table structures are used in conjunction with the EXECUTION STACK. They are the ADDRESS TABLE, the EXPRESSION TABLE and the CONSTANT TABLE.

The ADDRESS TABLE defines all variables. There are five (5) entries per variable:

1) ADDRESS - a thirty-two bit entry that holds an internal code that identifies each variable. This code is generated in sequence by the ALGOL-E compiler beginning with the first declared. This field will be zero if an ADDRESS TABLE entry defines an expression.

2) ADDTYPE - an eight bit entry that defines the type of the variable.

3) ADDVAL - a sixteen bit entry that holds the current class number assigned to the variable.

4) ADDCON - a sixteen bit entry that addresses the proper entry in the CONSTANT TABLE, if the current value of a variable is a constant.

5) ADDNAM - a sixteen bit index to the EXPRESSION TABLE if the entry defines an expression. In this case, the ADDRESS entry would be equal to zero.

The CONSTANT TABLE structure consists of four (4) elements that define each constant. The elements are:

1) CONSYM - a character string representation of the constant.

2) CONTYPE - an eight bit entry that defines the type for the constant.

3) CONVAL a sixteen bit entry that holds the class number assigned to the constant.

4) CONINT - a thirty-two bit binary representation of the actual constant value.

The EXPRESSION TABLE (VTAB) is structured as a linear table with sequential expression entries made as they occur. A hash coded retrieval method is used to access the table resulting in rapid search for redundant expressions. The lower portion of VTAB up to HASHBASE is saved for the primary hash link. Access to this link is formed by

$$\text{INDICATOR} + \text{OPERATOR} + \text{OPERAND}_0 \text{ class number} + \text{OPERAND}_1 \text{ class number} + \ldots + \text{OPERAND}_n \text{ class number.}$$

The link address obtained will either hold a zero indicating no expression with this hashcode has been encountered, or an index to the most recent expression that hashed to this address. The INDICATOR in the hash formula is used to determine where the class numbers for the operands can be found. Specifically, the expression is located either in the EXECUTION STACK or in the EXPRESSION TABLE. An EXPRESSION TABLE entry is structured as:

WORD	CONTENT
0	Bits 16-31 hold the original Hash Link address. Bits 0-15 hold the operator code.
1	Hash code collision field which is zero (0) if no further expression links are present.
2	Class number for operand$_0$.
.	Class number for operand$_1$
.	(variable number of operands
.	allowed).
2 + I	Class number for resultant operand$_0$.
2 + I + 1	Class number for resultant
.	operand$_1$ (variable number of
.	resultant operands allowed).
.	
2 + I + J + 1	Master chain pointer links all entries. Used for bookkeeping.

Additionally, the upper portions of the ADDRESS TABLE and EXPRESSION TABLE are used to hold the P_c for a block, as will be further explained in Section II.C.2.c.

The meta-execution operates by loading Polish operands onto the EXECUTION STACK and setting indexes to their respective ADDRESS TABLE entries. When an operator is detected in the Polish sequence, the EXPRESSION TABLE is searched for an identical entry using the hashcode chain. In this manner, common subexpressions are easily detected.

Because of the link from the ADDRESS TABLE to the CONSTANT TABLE, the detection of constant propagation is a simple matter of bit checking on the operands of an expression. If all operands are in fact constants, a propagated constant has been detected.

The application of simplifying formal identities is done by a comparison against a table of simplification rules generated by CSFG for each expression. Both propagated constants and simplification detection are accomplished by a call to the procedure SIMPLIFY by the BBA. The generated constant propagation case statement shown in TABLE III, is located in the procedure PROPAGATE which is invoked when a propagated constant is detected by the procedure SIMPLIFY. PROPAGATE then computes the propagated constant as specified by the operator. The constant value is then entered into the CONSTANT TABLE if it does not already exist.

As possible code transformations are detected, they are also flagged in the table OPTIM_TYPE which has a parallel entry for each code word. The entries in the OPTIM_TYPE TABLE are coded as:

CODE	MEANING
0	no transformation present
1	simplification possible
2	propagated constant
3	common subexpression
4	common subexpression that links to a propagated constant.

Additionally, an optimization entry is saved for future reference at the extreme top of the EXPRESSION TABLE (VTAB). The address of this optimization entry is entered into the ADDR field of the intermediate code for use by later passes. The form of this entry is:

VTAB LOCATION	CONTENTS
VTOPR	operand$_0$ class number
.	(variable number of
.	operands),
.	
VTOPR - X	code location at which an identical expression can be found. Zero (0) if not applicable,
VTOPR - X - 1	resultant operand class number,

where VTOPR is the current pointer to the last entry in the top of VTAB. Upon subsequent passes for a block, this

optimization expression entry will be updated as necessary. If a previous optimization does not recur on a subsequent pass, the indicator in OPTIM_TYPE is reset to zero (0), but the optimization expression entry remains linked.

During the BBA, the tables generated by the CSFG come into use. The linkage between tables is in some cases rather complex and needs clarification for a complete understanding of the CSF.

In order to identify the operator codes in the code file, two (2) tables are used:

1) OPRANGE
2) OPCODES.

OPRANGE is a table that is indexed by the OPCODE field in the code file. Each entry addresses the first character for a particular OPCODE in the character string called OP_CODES. Figure 6 illustrates this relationship.

To assist in the processing of expressions, five (5) tables are used:

1) OP_DEGL
2) OP_DEGR
3) OP_INDEX
4) OP_INFO
5) OP_STR.

OP_DEGL is indexed by OPCODE and contains the number of operands necessary for an operator.

OP_DEGR is indexed by OPCODE and contains the number of result operands that the operator will generate.

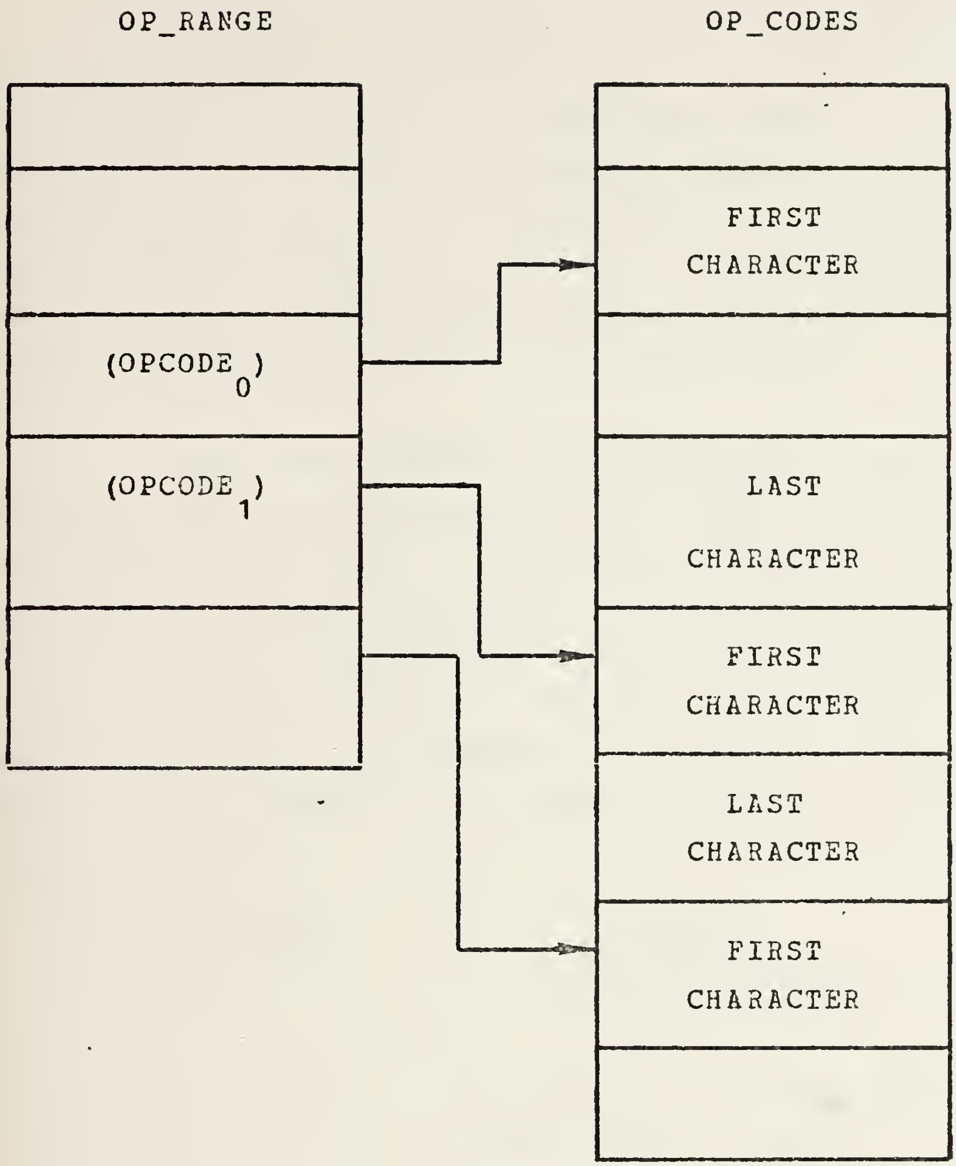

FIGURE 6

OP_RANGE TO OP_CODES RELATIONSHIP

OP_INDEX is indexed by OPCODE and points to the first entry in OP_INFO for an operand. OP_INFO holds the type code for each operand and resultant operand up to entry ninety-five (95) in the ALGOL-E implementation. As noted in Figure 7, more than one configuration of operand and resultant operand types can exist for a particular operator. The OP_DEGL and OP_DEGR counts provide a means of breaking a multiple configuration into its exclusive parts.

For the ALGOL-E implementation, the OP_INFO TABLE above entry ninety-five (95) is used in conjunction with the SIMP_INDEX TABLE and the OP_STR TABLE to identify simplification rules. SIMP_INDEX is indexed by the operator code and indexes the first simplification rule in OP_INFO for that operator. Table OP_STR holds the string representation of the identity operands "1" and "0" for ALGOL-E. Each entry of OP_INFO is eight (8) bits as follows:

BIT	CONTENT
0	1 = identity operand 0 = variable identifier
1-7	if identity operand this field points to OP_STR for proper identity. If variable operand this field holds the type code.

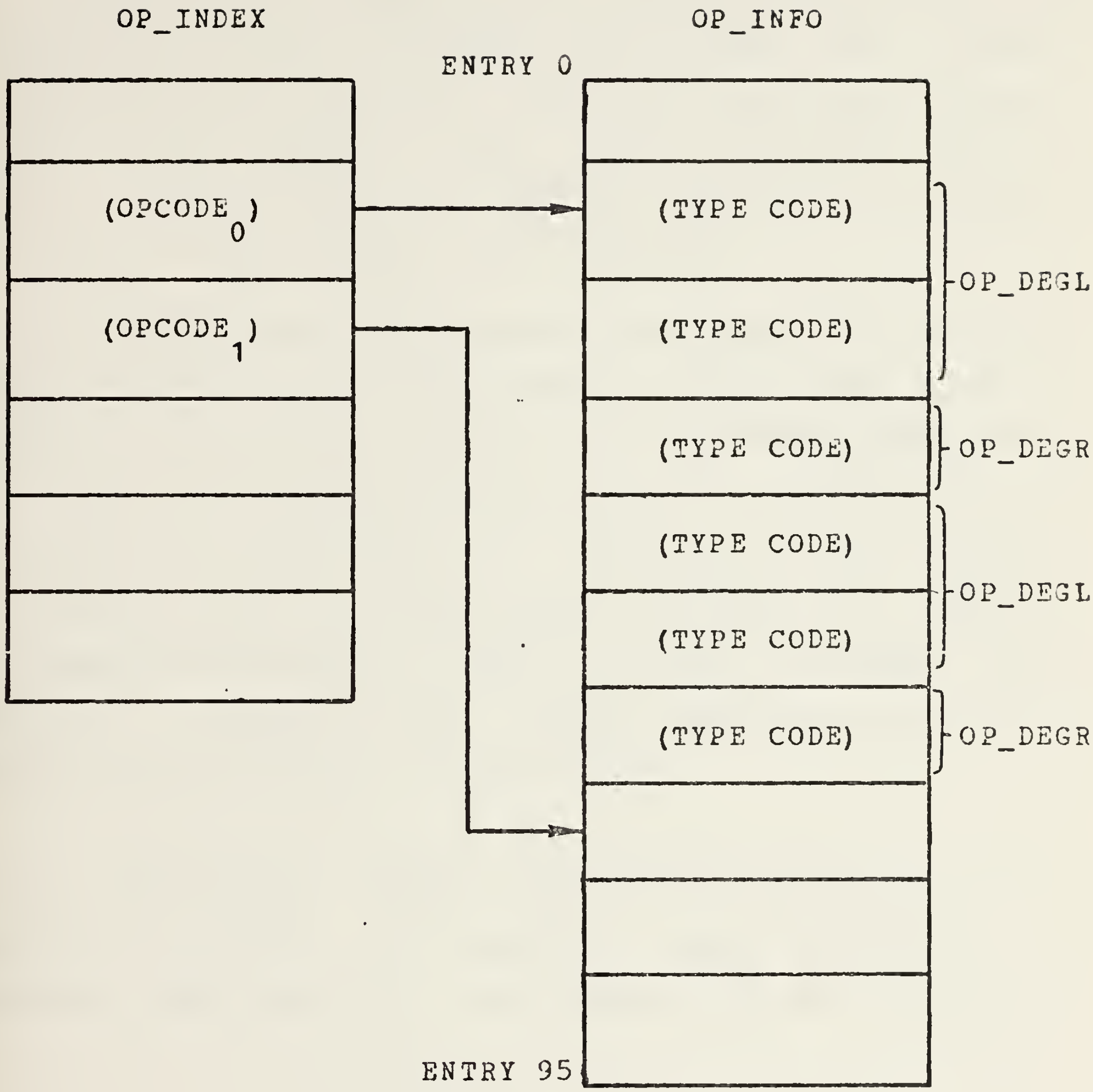

FIGURE_7

OP_INDEX TO OP_INFO RELATIONSHIP

Each entry of OP_INFO is a description of an operand or a resultant operand. Figure 8 depicts the Simplification Link Structure within OP_INFO. Note that more than one rule can exist for an operator in OP_INFO. A link from rule to rule is present, and OP_DEGR and OP_DEGL are used to process each rule. The last link field contains zero (0) which serves as the rule terminator for an operator. When a rule is matched against an existing expression, a simplification has been found.

The BBA sequentially processes the Polish elements beginning with the block's first word address provided by the CFA. Since the BBA processes only basic blocks, it returns control to the CFA upon encountering a branch operator or an ENT type variable. Any XIT operands encountered by the BBA result in entries on the CONTROL STACK with the ADDR field specifying the first word address of the immediate successor block. An ENT variable also results in a CONTROL STACK entry beginning at the next sequential instruction in the code file.

Because of constant propagation, the BBA is able to alter the actual block flow of the program if a propagated constant is detected while processing a conditional branch operator. For example, given the program

```
A := 5;
B := 10;
IF A > B THEN
C := 6;
D := 7;
```

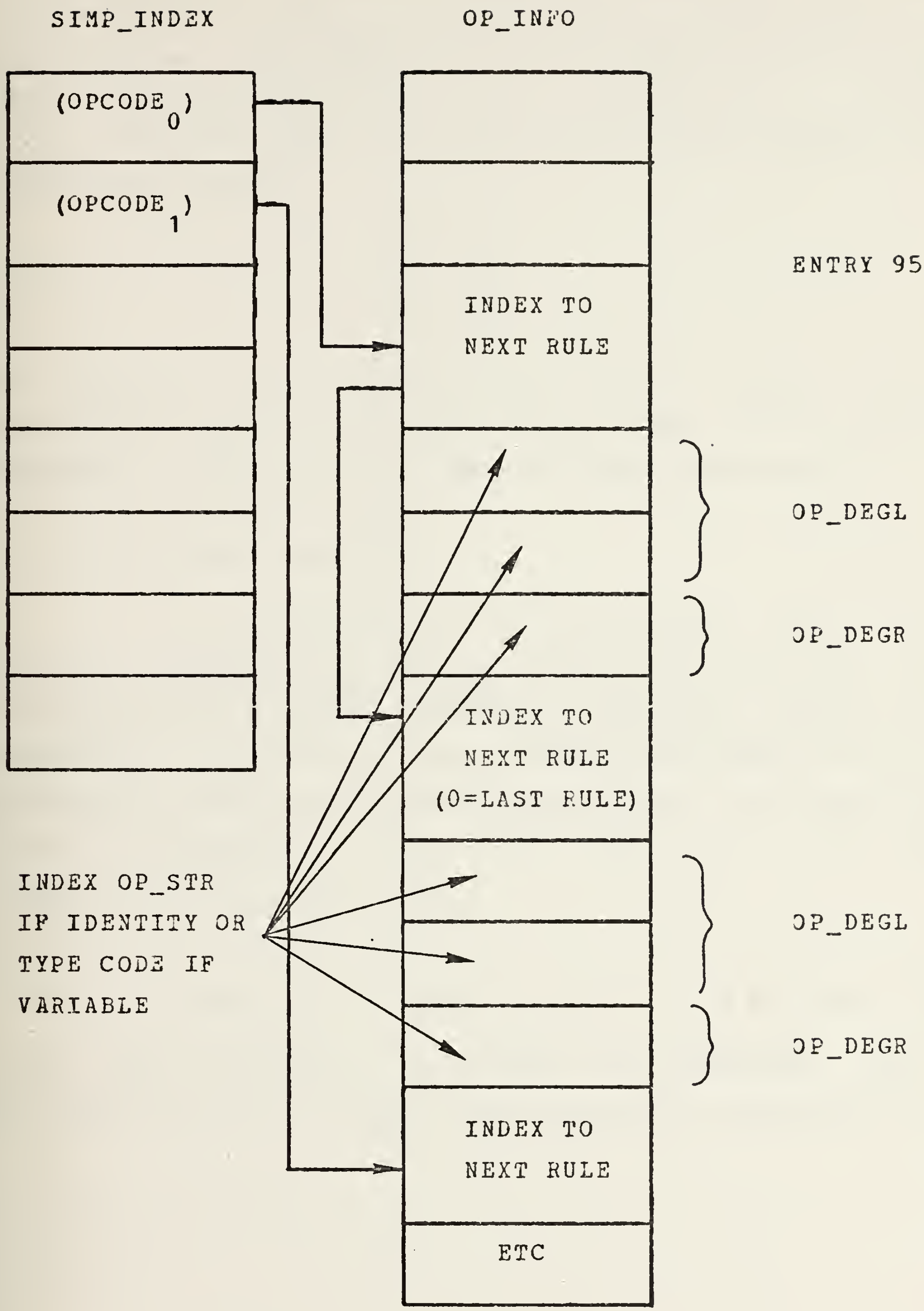

FIGURE 8

SIMPLIFICATION LINK STRUCTURE

the test of A > B would obviously always be false. When this condition is detected, only one of the XIT variables generated by the compiler would be placed on the CONTROL STACK. This can result in entire sections of a program never being traversed. The flow of the above example would effectively become

```
A := 5;
B := 10;
D := 7;
```

This facility can be controlled by a toggle called BRANCHTOG. If it is false, both exits will always be processed. Currently it is set to a true condition.

c. Meet Operation

The meet operation is accomplished from within procedure PERFORM_MEET_OP which is called by the CFA. Each call results in a single immediate successor block being processed. A true or false response indicates to the CFA whether the immediate successor block is to be marked for future processing. That is, if $P_c \leq P_i$, then PERFORM_MEET_OP returns false.

The P_c for a block is saved via the STATES TABLE in the upper portions of the ADDRESS and EXPRESSION TABLES. The STATES TABLE is a linear table which is structured as:

WORD	CONTENT
0	0
	always zero
1	0
2	number of states for block #1 (index to ADDRESS table for each state)
.	
.	.
.	
2 + I + 1	0
2 + I + 2	0
2 + I + 3	number of states for block #2
etc	etc

where a "state" is a single variable or expression description. The block header for each block points to its respective STATES TABLE entry. The ADDRESS TABLE entry for the P_c is nearly identical to the format previously discussed; however, it is saved linearly from the top down instead of from the bottom up. The EXPRESSION TABLE entry is also slightly different. An ADDRESS TABLE entry corresponding to an expression entry will still have zero (0) in the ADDRESS entry. However, the ADDNAM pointer will address the top of VTAB instead of the bottom. In VTAB, only the operands for the expression will be found. Note, finally, that the complete state of any one block is distinct from the other blocks; that is, no ADDRESS TABLE state entry is shared by blocks.

An important point in understanding the structure of the EXPRESSION TABLE (VTAB) is that it holds three separate entries. They are:

1) current expressions for the block being processed,
2) information on detected transformations, and
3) operands for the saved current optimized pools of all blocks.

Only the current expression entries are hash coded, and reside in the lower part of VTAB. The other two entry types are found in the upper regions of VTAB.

Prior to calling PERFORM_MEET_OP, the CFA structures the lower ADDRESS TABLE so that it completely reflects the P_O of the last processed block. This is accomplished by moving any EXECUTION STACK entries onto the ADDRESS TABLE. The ADDRESS field will be a number representing the relative position of the entry on the EXECUTION STACK. The uppermost bit (bit 15) of the ADDRESS field is set to identify the entry as originating from the EXECUTION STACK. Therefore, the meet operation need only look at the lower ADDRESS TABLE for the P_O and the upper ADDRESS TABLE specified by the STATES TABLE for the P_C of the immediate successor block.

The meet operation is accomplished in three (3) steps:

1) Initialize
2) Process all simple variables
3) Process all expressions.

Two (2) tables, the CLASS OCCURRENCE LIST and the CLASS REFINEMENT TABLE, are the primary structures used to accomplish the meet operation as developed by Kildall [28].

The initialization stage builds the CLASS OCCURRENCE LIST based on the P_c of the block to be processed. This list is simply a count of the occurrence of each class number that is assigned to a simple variable or an expression.

After the CLASS OCCURRENCE LIST is built, all simple variables are processed from the P_c of the immediate successor block. The P_o is searched for each identifier in the P_c. If the identifier exists, the class numbers from the two (2) pools are entered into the CLASS REFINEMENT LIST. If the class numbers are not the same, a new "refined class number" is assigned to the identifier. The identifier is then placed in the P_i with the refined class number. As each identifier is processed from the P_c, its class occurrence is decremented in the CLASS OCCURRENCE LIST.

After all simple variables have been processed, each expression in the P_c is considered. When an unprocessed expression is found, a check is made to ensure that the the CLASS OCCURRENCE LIST for all its operand class numbers are zero (0). If they are not, the expression is not processed until they are, with a single exception. Simplification rules such as

A + 0 = A

cause both (A + 0) and A to appear in the same class. When this occurs, (A + 0) is treated the same as A.

If an expression can be processed, its resultant class number count is decremented in the CLASS OCCURRENCE

LIST. A list of candidate expressions to consider in the P_o is formed by examining the CLASS REFINEMENT LIST. If an expression in the P_c has operands OP_1 and OP_2, a search of the CLASS REFINEMENT LIST will be made to identify class numbers in the P_o that are equivalent to OP_1 and OP_2. For example, if an output pool is

$$P_o = \{\underset{(1)}{A} \mid \underset{(2)}{B} \mid \underset{(3)}{A + B,\ B + A}\}$$

and a current pool is

$$P_c = \{\underset{(5)}{A} \mid \underset{(6)}{B} \mid \underset{(4)}{A + B,\ B + A}\}$$

where each class is labeled with a class number. The CLASS OCCURRENCE LIST would be:

NUMBER OF OCCURRENCES	CLASS NUMBER
1	4
1	5
1	6

After processing simple variables, the CLASS OCCURRENCE LIST would be:

NUMBER OF OCCURRENCES	CLASS NUMBER
1	4

and the CLASS REFINEMENT TABLE would be

P_o CLASS NUMBER	P_c CLASS NUMBER	REFINED CLASS NUMBER
1	5	7
2	6	8

Since identifiers A and B were both found, the P_i to this point would be

$$P_i = \{ A \mid B \}.$$

(7) (8)

Upon searching the CLASS OCCURRENCE LIST for the operand class numbers of the expression (A + B) in the P_c, none will be found; therefore, the expression can be processed. The resultant class number, 4, is deleted from the CLASS OCCURRENCE LIST. Using the CLASS REFINEMENT TABLE a list of candidate expressions to search for in the P_o is constructed. This list is developed by taking the operands of the expression in the P_c and searching the CLASS REFINEMENT TABLE for matches in the P_o. For example, a class number of 1 in the P_o is equivalent to a class number of 5 in the P_c and a class number of 2 in the P_o is equivalent to a class number of 6 in the P_c. Therefore, the expressions to be searched for would be

<u>P_o EXPRESSION LIST</u>

(5) + (6)

(6) + (5)

since the operator "+" is commutative. The P_o is then searched for an occurrence of these expressions. If found, the resulting class numbers of the P_c and P_o expressions are entered in the CLASS REFINEMENT TABLE in the same manner as for an identifier.

P_O CLASS NUMBER	P_C CLASS NUMBER	REFINED CLASS NUMBER
1	5	7
2	6	8
3	4	9

An expression entry can then be made in the P_i using the refined class numbers for both operands and resultant operands. The P_i would then be

$$P_i = \{ \underset{(7)}{A} \mid \underset{(8)}{B} \mid \underset{(9)}{A + B, B + A} \}.$$

After all expressions have been processed, the CLASS OCCURRENCE LIST will be empty. If an expression or identifier in the P_c is not found in the P_o, the current state of P_c has been altered and PERFORM_MEET_OP will return true, indicating the block must be traversed again.

A problem with the handling of constants occurs if an identifier or expression is a constant in the P_c and is present in the P_o but its value is not a constant in the P_o. The state has changed in this situation, thus requiring a true response from PERFORM_MEET_OP.

As each identifier and expression is processed, the P_c area at the top of the ADDRESS TABLE is updated. This is the P_i that is being formed, which is to be loaded onto the lower ADDRESS TABLE by the CFA before calling the BBA to process the block. At the completion of the CSF, the

P_C held within the upper ADDRESS TABLE is the P_F (i.e., MOP) for each block.

d. Output Formats

A number of table formats have already been discussed; however, further output format definitions for the equivalence classes and the final optimization information is necessary.

The state elements have one of three formats:

1) A (X, Y, Z)
2) E (X, Y, Z)
3) C (X, Y, Z).

Form A(X, Y, Z) refers to a simple identifier and

X is the identifier code number,

Y is the class number of the constant if identifier X is equal to a constant, and

Z is the class number assigned the identifier X in the ADDRESS TABLE.

Form E(X, Y, Z) refers to an EXECUTION STACK entry and

X is the relative location of the entry from the top of the EXECUTION STACK. Y and Z are the same as for a simple identifier.

Form C(X, Y, Z) refers to an expression entry and

X represents the class numbers of all operands separated by colons,

Y is the class number of the constant

if the resultant class number is equal to a constant, and

Z is the class number of the resultant operand.

Another format for expressions is used when the expression stack is output at the end of processing each block. Its form is:

C (X, Y, Z)

where

X is the string representation of the expression operator,

Y is the same as for X in state expressions, and

Z is the same as for Z in state expressions.

The final optimization results are output in the form

Optimization at X
$(a_1, a_2, \ldots, a_k \mid b, c)$

where

X is the location in the intermediate code file where the optimization was detected,

$a_1, a_2, \ldots, a_k$ are the class numbers of all operands,

b is the address of a previous expression that is identical, if one exists, and

c is the resultant class number of the expression.

In all of the above formats, an unknown element will be

printed as "*."

The block headers are output at the end of the optimization process. The format for each block is:

LINE	CONTENT
1	block number
2	first and last word addresses within the intermediate code file for the block
3	count for the number of times the block was traversed
4	block number of an immediate predecessor block
4 + I	(variable number of predecessor blocks possible).

The intermediate code file is also printed, and the C field indicates the actual location of any optimization information. The OPTIM_TYPE TABLE is output concurrent with the code file to indicate the optimization type.

Several examples of CSF output are enclosed in which different block selection methods were used as indicated on the output. TEST program number six (6) has all four block selection methods listed so that a comparison of the CSF flow using different block selection methods can be seen.

TABLE V is the CSF output from the source program shown in TABLE IV with $TABLE selected. The different tables are identified as they occur.

TABLE V

TABLE TRACE OUTPUT FROM CSF

```
***CODE SYNTHESIS FILTER***

BLOCK SELECTION METHOD WILL BE STEEPEST DESCENT (MINIMUM CURRENT POOL)

***C O M P L E T E   T A B L E   D U M P***
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

***T A B L E   D U M P*** AT LOCATION = 0

CONSTANTS <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
CONSYM    |2   |3
CONTYPE   |INT |INT
CONVAL    |46  |47
CONINT    |2   |3
```

CONSTANT TABLE FROM CSFE

```
ADDR TABL <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)
```

SIMPLE VARIABLES STACK IS EMPTY

```
VALUE STACK (TOP AT 127)
```

EXPRESSION STACK IS EMPTY

```
CONT STK  <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
BLOCK#    |0
ENTRY     |0
```

SET TO PROCESS BLOCK #1
STARTING AT LOCATION 0

```
EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)
```

EXECUTION STACK IS EMPTY

**** FORMATTED INTERMEDIATE CODE DUMP ****

LOC	OFF SET	OP CODE	CNS	ADR /ETC	RAW CODE	OPTIMIZATION TYPE
0	0	ENT	C	46	0003802E	
1	0	TOGGLE		257	00080101	
2	0	TOGGLE		258	00080102	
3	0	LOC		3	00020003	
4	0	INT		1	00050001	
5	0	STO		0	00280000	
6	0	DEL		0	002A0000	
7	0	LOC		4	00020004	
8	0	INT		2	00050002	
9	0	STO		0	00280000	
10	0	DEL		0	002A0000	
11	0	LOC		1	00020001	
12	0	INT		0	00050000	
13	C	STD		0	00290000	
14	0	LOC		2	00020002	
15	0	INT		0	00050000	
16	0	STD		0	00290000	
17	0	ENT		1	00030001	
18	0	LOC		1	00020001	
19	0	LOD		0	00270000	
20	0	LOC		2	00020002	
21	0	LOD		0	00270000	
22	0	GTR		0	00210000	
23	0	XIT		37	00040025	
24	0	XIT		26	0004001A	
25	0	BSC		0	002F0000	
26	0	ENT		1	00030001	
27	0	LOC		2	00020002	
28	0	LOC		3	00020003	
29	0	LOD		0	00270000	
30	0	LOC		4	00020004	
31	0	LOD		0	00270000	
32	0	ADD		0	00100000	
33	0	STO		0	00280000	
34	0	DEL		0	002A0000	
35	0	XIT		17	00040011	
36	0	BRS		0	002F0000	
37	0	ENT		1	00030001	
38	0	LOC		5	00020005	
39	0	LOC		4	00020004	
40	0	LOD		0	00270000	
41	0	LOC		3	00020003	
42	0	LOD		0	00270000	
43	0	ADD		0	00100000	
44	0	STO		0	00280000	
45	0	DEL		0	002A0000	

INITIAL ENT IS FORCED BY THE INITIALIZATION ROUTINES

TABLE V (CON'T)

```
BASIC BLOCK #1
      BEGINNING AT 0, ENDING AT 0                    BLOCK HEADER
      BLOCK TRAVERSED 0 TIMES
      0 REFERENCES:

***E N D   O F   C O M P L E T E   T A B L E   D U M P***

B+2     0    |TOGGLE      |  |258    | 000B0102  |      TRACE IN BBA AT LOCATION 2

***T A B L E   D U M P*** AT LOCATION = 2

 INPUT POOL FOR  BLOCK# 1 IS:
(NULL)                                               BLOCK #1 HAS A NULL INPUT POOL

ADDR TABL <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

VALUE STACK (TOP AT 127)


CONT STK  <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)


EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

B+3     0    |LOC         |  |3      | 00020003  |

***T A B L E   D U M P*** AT LOCATION = 3
ADDR TABL <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
ADDRESS   |3
ADDTYPE   |NULL
ADDVAL    |0                           LOC OPERATOR HAS RESULTED IN A SIMPLE
ADDCON    |0                           VARIABLE ENTRY BEING MADE IN ADDRESS TABLE


EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |N
EXTYPE    |LOC                         EXECUTION STACK HOLDS ONE ENTRY OF TYPE LOC
EXADD     |3                           WHICH IS REFERENCING SIMPLE VARIABLE 3 (C)
EXVAL#    |0

B+4     0    |INT         |  |1      | 00050001  |

***T A B L E   D U M P*** AT LOCATION = 4
EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |N   |Y
EXTYPE    |LOC |INT                    THE CONSTANT OF 2 IS NOW ON THE TOP OF THE
EXADD     |3   |1                      EXECUTION STACK.  NOTE THE CONSTANT INDICATOR
EXVAL#    |0   |46                     AND THE CLASS NUMBER.

B+5     0    |STO         |  |0      | 00280000  |

***T A B L E   D U M P*** AT LOCATION = 5
ADDR TABL <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
ADDRESS   |3
ADDTYPE   |INT                         THE VARIABLE 3 IS NOW SET TO POINT AT THE
ADDVAL    |46                           CONSTANT 2.
ADDCON    |1        POINTS TO CONSTANT TABLE

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |Y
EXTYPE    |INT
EXADD     |1
EXVAL#    |46

B+6     0    |DEL         |  |0      | 002A0000  |

***T A B L E   D U M P*** AT LOCATION = 6
EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)
```

TABLE V (CON'T)

```
B+7     0    |LOC           | |4       | 00020004  |

***T A B L E   D U M P*** AT LOCATION = 7
ADDR TABL <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
ADDRESS   |3    |4
ADDTYPE   |INT  |NULL
ADDVAL    |46   |0
ADDCON    |1    |0

EXEC STAC <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
EXCON     |N
EXTYPE    |LOC
EXADD     |4
EXVAL#    |0

B+8     0    |INT           | |2       | 00050002  |

***T A B L E   D U M P*** AT LOCATION = 8
EXEC STAC <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
EXCON     |N    |Y
EXTYPE    |LOC  |INT
EXADD     |4    |2
EXVAL#    |0    |47

B+9     0    |STO           | |0       | 00280000  |

***T A B L E   D U M P*** AT LOCATION = 9
ADDR TABL <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
ADDRESS   |3    |4
ADDTYPE   |INT  |INT
ADDVAL    |46   |47
ADDCON    |1    |2

EXEC STAC <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
EXCON     |Y
EXTYPE    |INT
EXADD     |2
EXVAL#    |47

B+10    0    |DEL           | |0       | 002A0000  |

***T A B L E   D U M P*** AT LOCATION = 10
EXEC STAC <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
(EMPTY)

B+11    0    |LOC           | |1       | 00020001  |

***T A B L E   D U M P*** AT LOCATION = 11
ADDR TABL <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
ADDRESS   |3    |4    |1
ADDTYPE   |INT  |INT  |NULL
ADDVAL    |46   |47   |0
ADDCON    |1    |2    |0

EXEC STAC <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
EXCON     |N
EXTYPE    |LOC
EXADD     |1
EXVAL#    |0

B+12    0    |INT           | |0       | 00050000  |

***T A B L E   D U M P*** AT LOCATION = 12
EXEC STAC <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
EXCON     |N    |N
EXTYPE    |LOC  |INT
EXADD     |1    |0
EXVAL#    |0    |48

B+13    0    |STO           | |0       | 00290000  |
```

```
***T A B L E   D U M P*** AT LOCATION = 13
ADDR TABL <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
ADDRESS   |3    |4    |1
ADDTYPE   |INT  |INT  |INT
ADDVAL    |46   |47   |48
ADDCON    |1    |2    |0

EXEC STAC <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
(EMPTY)

B+14    0    |LOC          | |2       | 00020002  |

***T A B L E   D U M P*** AT LOCATION = 14
ADDR TABL <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
ADDRESS   |3    |4    |1    |2
ADDTYPE   |INT  |INT  |INT  |NULL
ADDVAL    |46   |47   |48   |0
ADDCON    |1    |2    |0    |0

EXEC STAC <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
EXCON     |N
EXTYPE    |LOC
EXADD     |2
EXVAL#    |0

B+15    0    |INT          | |0       | 00050000  |

***T A B L E   D U M P*** AT LOCATION = 15
EXEC STAC <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
EXCON     |N    |N
EXTYPE    |LOC  |INT
EXADD     |2    |0
EXVAL#    |0    |49

B+16    0    |STD          | |0       | 00290000  |

***T A B L E   D U M P*** AT LOCATION = 16
ADDR TABL <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
ADDRESS   |3    |4    |1    |2
ADDTYPE   |INT  |INT  |INT  |INT
ADDVAL    |46   |47   |48   |49
ADDCON    |1    |2    |0    |0

EXEC STAC <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
(EMPTY)

B+17    0    |ENT          | |1       | 00030001  |
 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 2 IS:
 A(2,*,49)  A(1,*,48)  A(4,47,47)  A(3,46,46)  <----------  THE INITIAL CURRENT
                                                            POOL FOR BLOCK #2 IS THE
                                                            OUTPUT POOL FROM BLOCK #1

***T A B L E   D U M P*** AT LOCATION = 17
CONT STK  <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
BLOCK#    |1
ENTRY     |17 <----------  BLOCKS TO BE PROCESSED LIST HAS A SINGLE
                           ENTRY FROM BLOCK #1 TO LOCATION 17.

EXEC STAC <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
(EMPTY)

 CONTROL AT 17, FROM BLK 1 TO BLK 2 (PASS 1)
C+17    0    |ENT          |C|50      | 00038032  | <---- CFA PASSES CONTROL FROM BLOCK #1 TO
                                                          BLOCK #2.

***T A B L E   D U M P*** AT LOCATION = 17

 INPUT POOL FOR  BLOCK# 2 IS:
A(2,*,49)  A(1,*,48)  A(4,47,47)  A(3,46,46) <---- INPUT POOL FOR BLOCK #2..STATE IS FOUR SIMPLE
                                                   VARIABLES, TWO OF WHICH ARE CONSTANTS.
ADDR TABL <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
ADDRESS   |3    |4    |1    |2
ADDTYPE   |INT  |INT  |INT  |INT
ADDVAL    |46   |47   |48   |49
ADDCON    |1    |2    |0    |0
```

TABLE V (CON'T)

```
VALUE STACK (TOP AT 127)          EXPRESSION STACK IS EMPTY

CONT STK  <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

B+18    0    |LOC          | |1      | 00020001  |

***T A B L E   D U M P*** AT LOCATION = 18
EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |N
EXTYPE    |LOC
EXADD     |1
EXVAL#    |0

B+19    0    |LOD          | |0      | 00270000  |

***T A B L E   D U M P*** AT LOCATION = 19
EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |N
EXTYPE    |INT
EXADD     |0
EXVAL#    |48

B+20    0    |LOC          | |2      | 00020002  |

***T A B L E   D U M P*** AT LOCATION = 20
EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |N    |N
EXTYPE    |INT  |LOC
EXADD     |0    |2
EXVAL#    |48   |0

B+21    0    |LOD          | |0      | 00270000  |

***T A B L E   D U M P*** AT LOCATION = 21
EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |N    |N
EXTYPE    |INT  |INT
EXADD     |0    |0
EXVAL#    |48   |49

B+22    0    |GTR          | |0      | 00210000  |

***T A B L E   D U M P*** AT LOCATION = 22     AN EXPRESSION ENTRY IS MADE..WHILE A GTR B DO
VALUE STACK (TOP AT 133)                       OPERANDS OF 48 AND 49 AND A RESULT OF 50.
C(GTR 48 49 |50)

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |N
EXTYPE    |BOOL
EXADD     |0
EXVAL#    |50

B+23    0    |XIT          | |37     | 00040025  |

***T A B L E   D U M P*** AT LOCATION = 23
CONT STK  <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
BLOCK#    |2
ENTRY     |37

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |N
EXTYPE    |BOOL
EXADD     |0
EXVAL#    |50
```

TABLE V (CON'T)

```
B+24    0    |XIT         | |26      | 0004001A  |

***T A B L E   D U M P*** AT LOCATION = 24
CONT STK  <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
BLOCK#    |2   |2
ENTRY     |37  |26
```

BLOCK #2 HAS GENERATED TWO IMMEDIATE SUCCESSOR BLOCKS.

```
EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |N
EXTYPE    |BOOL
EXADD     |0
EXVAL#    |50

B+25    0    |BSC         | |0       | 002F0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 3 IS:
 C(48:49,*,50)  A(2,*,49)  A(1,*,48)  A(4,47,47)  A(3,46,46)

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 4 IS:
 C(48:49,*,50)  A(2,*,49)  A(1,*,48)  A(4,47,47)  A(3,46,46)
```

CURRENT POOL FOR BOTH BLOCK #3 AND #4 HAS BEEN SET TO THE OUTPUT POOL FROM BLOCK #2.

```
***T A B L E   D U M P*** AT LOCATION = 25
ADDR TABL <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
ADDRESS   |3   |4   |1   |2   |0
ADDTYPE   |INT |INT |INT |INT |GTR
ADDVAL    |46  |47  |48  |49  |50
ADDCON    |1   |2   |0   |0   |0

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

 CONTROL AT 26, FROM BLK 2 TO BLK 4 (PASS 1)
C+26    0    |ENT         |C|60    | 0003803C  |
```

CFA PASSES CONTROL FROM BLOCK #2 TO BLOCK #4.

```
***T A B L E   D U M P*** AT LOCATION = 26

 INPUT POOL FOR  BLOCK# 4 IS:
 C(48:49,*,50)  A(2,*,49)  A(1,*,48)  A(4,47,47)  A(3,46,46)
```

INPUT POOL FOR BLOCK #4 IS FOUR SIMPLE VARIABLES AND ONE EXPRESSION.

```
ADDR TABL <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
ADDRESS   |3   |4   |1   |2
ADDTYPE   |INT |INT |INT |INT
ADDVAL    |46  |47  |48  |49
ADDCON    |1   |2   |0   |0

VALUE STACK (TOP AT 133)
C(GTR 48 49 |50)

CONT STK  <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
BLOCK#    |2
ENTRY     |37

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

B+27    0    |LOC         | |2       | 00020002  |

***T A B L E   D U M P*** AT LOCATION = 27
EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |N
EXTYPE    |LOC
EXADD     |2
EXVAL#    |0

B+28    0    |LOC         | |3       | 00020003  |

***T A B L E   D U M P*** AT LOCATION = 28
EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |N   |N
EXTYPE    |LOC |LOC
EXADD     |2   |3
EXVAL#    |0   |0
```

```
B+29    0   |LOD          | |0      | 00270000  |

***T A B L E   D U M P*** AT LOCATION = 29
EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |N   |Y
EXTYPE    |LOC |INT
EXADD     |2   |1
EXVAL#    |0   |46

B+30    0   |LOC          | |4      | 00020004  |

***T A B L E   D U M P*** AT LOCATION = 30
EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |N   |Y   |N
EXTYPE    |LOC |INT |LOC
EXADD     |2   |1   |4
EXVAL#    |0   |46  |0

B+31    0   |LOD          | |0      | 00270000  |

***T A B L E   D U M P*** AT LOCATION = 31
EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |N   |Y   |Y
EXTYPE    |LOC |INT |INT
EXADD     |2   |1   |2
EXVAL#    |0   |46  |47

B+32    0   |ADD          | |0      | 00100000  |

***T A B L E   D U M P*** AT LOCATION = 32
CONSTANTS <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
CONSYM    |2   |3   |5
CONTYPE   |INT |INT |INT
CONVAL    |46  |47  |52
CONINT    |2   |3   |5
                                   NEW EXPRESSION ADDED
                                       B := C + D;
VALUE STACK (TOP AT 139)
C(ADD 46 47 |52) C(GTR 48 49 |50)

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |N   |Y
EXTYPE    |LOC |INT
EXADD     |2   |3
EXVAL#    |0   |52

B+33    0   |STO          | |0      | 00280000  |

***T A B L E   D U M P*** AT LOCATION = 33
ADDR TABL <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
ADDRESS   |3   |4   |1   |2   |2
ADDTYPE   |INT |INT |INT |INT |INT
ADDVAL    |46  |47  |48  |49  |52
ADDCON    |1   |2   |0   |0   |3

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |Y
EXTYPE    |INT
EXADD     |3
EXVAL#    |52

B+34    0   |DEL          | |0      | 002A0000  |

***T A B L E   D U M P*** AT LOCATION = 34
EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

B+35    0   |XIT          | |17     | 00040011  |
```

TABLE V (CON'T)

```
***T A B L E   D U M P*** AT LOCATION = 35
CONT STK  <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
BLOCK#    |2    |4
ENTRY     |37   |17

EXEC STAC <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
(EMPTY)

B+36   0   |BRS         | |0     | 002E0000  |
BEFORE THE MEET OPERATION:
```

MEET OPERATION IS NOW PERFORMED USING THE FIRST BLOCK ON THE CONTROL LIST

```
 OUTPUT POOL FROM BLOCK# 4 IS:
 C(48:49,*,50)  C(46:47,*,52)  A(2,52,52)  A(1,*,48)  A(4,47,47)  A(3,46,46)

 CURRENT POOL FOR BLOCK# 2 IS:
 A(2,*,49)  A(1,*,48)  A(4,47,47)  A(3,46,46)

CLASS OCCURENCE LIST:

ADDR        CLASS NUMBER      NUMBER OF OCCURRENCES
1                49                 1
2                48                 1
3                47                 1
4                46                 1
```

FOUR CLASS NUMBERS ARE IN THE OCCURENCE LIST

```
AFTER PROCESSING SIMPLE VARIABLES:
CLASS OCCURENCE LIST:
(EMPTY)
```

SIMPLE VARIABLES HAVE EXHAUSTED THE CLASS OCCURENCE LIST

```
CLASS REFINEMENT TABLE:

LOC         NEW VALUE#       CURRENT VALUE#       INPUT VALUE#

1               53               49                   52
2               48               48                   48
3               47               47                   47
4               46               46                   46
```

NEW CLASS NUMBERS ASSIGNED ONLY IF CLASS NUMBERS DIFFER

```
AFTER PROCESSING ALL EXPRESSIONS:
CLASS OCCURENCE LIST:
(EMPTY)

CLASS REFINEMENT TABLE:

LOC         NEW VALUE#       CURRENT VALUE#       INPUT VALUE#

1               53               49                   52
2               48               48                   48
3               47               47                   47
4               46               46                   46

AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL
```

BLOCK #2 WILL NOT BE PROCESSED BECAUSE ITS CURRENT POOL IS SAME

```
***T A B L E   D U M P*** AT LOCATION = 36
ADDR TABL <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
ADDRESS   |3    |4    |1    |2    |2    |0    |0
ADDTYPE   |INT  |INT  |INT  |INT  |INT  |ADD  |GTR
ADDVAL    |46   |47   |48   |49   |52   |52   |50
ADDCON    |1    |2    |0    |0    |3    |0    |0

EXEC STAC <1>   <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
(EMPTY)

 CONTROL AT 37, FROM BLK 2 TO BLK 3 (PASS 1)
C+37   0   |ENT         |C|55    | 00038037  |
```

CFA PASSES CONTROL FROM BLOCK #2 TO BLOCK #3.

```
***T A B L E   D U M P*** AT LOCATION = 37

 INPUT POOL FOR  BLOCK# 3 IS:
 C(48:49,*,50)  A(2,*,49)  A(1,*,48)  A(4,47,47)  A(3,46,46)
```

```
ADDR TABL <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
ADDRESS   |3   |4   |1   |2
ADDTYPE   |INT |INT |INT |INT
ADDVAL    |46  |47  |48  |49
ADDCON    |1   |2   |0   |0

VALUE STACK (TOP AT 133)
C(GTR 48 49 |50)

CONT STK  <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

B+38    0    |LOC         | |5      | 00020005  |

***T A B L E   D U M P*** AT LOCATION = 38
ADDR TABL <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
ADDRESS   |3   |4   |1   |2   |5
ADDTYPE   |INT |INT |INT |INT |NULL
ADDVAL    |46  |47  |48  |49  |0
ADDCON    |1   |2   |0   |0   |0

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |N
EXTYPE    |LOC
EXADD     |5
EXVAL#    |0

B+39    0    |LOC         | |4      | 00020004  |

***T A B L E   D U M P*** AT LOCATION = 39
EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |N   |N
EXTYPE    |LOC |LOC
EXADD     |5   |4
EXVAL#    |0   |0

B+40    0    |LOD         | |0      | 00270000  |

***T A B L E   D U M P*** AT LOCATION = 40
EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |N   |Y
EXTYPE    |LOC |INT
EXADD     |5   |2
EXVAL#    |0   |47

B+41    0    |LOC         | |3      | 00020003  |

***T A B L E   D U M P*** AT LOCATION = 41
EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |N   |Y   |N
EXTYPE    |LOC |INT |LOC
EXADD     |5   |2   |3
EXVAL#    |0   |47  |0

B+42    0    |LOD         | |0      | 00270000  |

***T A B L E   D U M P*** AT LOCATION = 42
EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
EXCON     |N   |Y   |Y
EXTYPE    |LOC |INT |INT
EXADD     |5   |2   |1
EXVAL#    |0   |47  |46

B+43    0    |ADD         | |0      | 00100000  |

***T A B L E   D U M P*** AT LOCATION = 43
VALUE STACK (TOP AT 139)
C(ADD 47 46 |52) C(GTR 48 49 |50)
```

```
EXEC STAC <1>  <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
EXCON     |N   |Y
EXTYPE    |LOC |INT
EXADD     |5   |3
EXVAL#    |0   |52

B+44    0    |STO          | |0        | 00280000  |

***T A B L E   D U M P*** AT LOCATION = 44
ADDR TABL <1>  <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
ADDRESS   |3   |4    |1    |2    |5
ADDTYPE   |INT |INT  |INT  |INT  |INT
ADDVAL    |46  |47   |48   |49   |52
ADDCON    |1   |2    |0    |0    |3

EXEC STAC <1>  <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
EXCON     |Y
EXTYPE    |INT
EXADD     |3
EXVAL#    |52

B+45    0    |DEL          | |0        | 002A0000  |

***T A B L E   D U M P*** AT LOCATION = 45
EXEC STAC <1>  <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
(EMPTY)


***T A B L E   D U M P*** AT LOCATION = 45
ADDR TABL <1>  <2>   <3>   <4>   <5>   <6>   <7>   <8>   <9>   <10> <11> <12> <13> <14>
ADDRESS   |3   |4    |1    |2    |5    |0    |0
ADDTYPE   |INT |INT  |INT  |INT  |INT  |ADD  |GTR
ADDVAL    |46  |47   |48   |49   |52   |52   |50
ADDCON    |1   |2    |0    |0    |3    |0    |0

                                   FINAL OPTIMIZATION RESULTS
##############################################################################################
----------------------------------------------------------------------------------------------

OPTIMIZATION AT 32
          (47,46|32,52)          CONSTANT PROPAGATION

OPTIMIZATION AT 43
          (47,46|32,52)          CONSTANT PROPAGATION
```

AT LOCATION 32 AN EXPRESSION WAS DETECTED AS A CONSTANT PROPAGATION

AT LOCATION 43 AN EXPRESSION IS IDENTICLE TO AN EXPRESSION AT LOCATION 32

TABLE V (CON'T)

**** FORMATTED INTERMEDIATE CODE DUMP ****

LOC	OFF SET	OP CODE	CNS	ADR /ETC	RAW CODE	OPTIMIZATION TYPE
0	0	ENT		46	0003002E	
1	0	TOGGLE		257	000B0101	
2	0	TOGGLE		258	000B0102	
3	0	LOC		3	00020003	
4	0	INT		1	00050001	
5	0	STO		0	00280000	
6	0	DEL		0	002A0000	
7	0	LOC		4	00020004	
8	0	INT		2	00050002	
9	0	STO		0	00280000	
10	0	DEL		0	002A0000	
11	0	LOC		1	00020001	
12	0	INT		0	00050000	
13	0	STD		0	00290000	
14	0	LOC		2	00020002	
15	0	INT		0	00050000	
16	0	STD		0	00290000	
17	0	ENT		50	00030032	
18	0	LOC		1	00020001	
19	0	LOD		0	00270000	
20	0	LOC		2	00020002	
21	0	LOD		0	00270000	
22	0	GTR		1024	00210400	
23	0	XIT		37	00040025	
24	0	XIT		26	0004001A	
25	0	BSC		0	002F0000	
26	0	ENT		60	0003003C	
27	0	LOC		2	00020002	
28	0	LOC		3	00020003	
29	0	LOD		0	00270000	
30	0	LOC		4	00020004	
31	0	LOD		2	00270000	
32	0	ADD	C	1016	001083F8	CONSTANT PROPAGATION
33	0	STO		0	00280000	
34	0	DEL		0	002A0000	
35	0	XIT		17	00040011	
36	0	BRS		0	002E0000	
37	0	ENT		55	00030037	
38	0	LOC		5	00020005	
39	0	LOC		4	00020004	
40	0	LOD		0	00270000	
41	0	LOC		3	00020003	
42	0	LOD		0	00270000	
43	0	ADD	C	1012	001083F4	CONSTANT PROPAGATION
44	0	STO		0	00280000	
45	0	DEL		0	002A0000	

OPTIMIZATIONS ARE FLAGGED IN
IN THE C-FIELD

```
      FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

 FINAL POOL FOR  BLOCK# 1 IS:                           ACTUALLY THE MOP FOR EACH BLOCK.
(NULL)


 FINAL POOL FOR  BLOCK# 2 IS:
 A(2,*,53)  A(1,*,48)  A(4,47,47)  A(3,46,46)

 FINAL POOL FOR  BLOCK# 3 IS:
 C(48:49,*,50)  A(2,*,49)  A(1,*,48)  A(4,47,47)  A(3,46,46)

 FINAL POOL FOR  BLOCK# 4 IS:
 C(48:49,*,50)  A(2,*,49)  A(1,*,48)  A(4,47,47)  A(3,46,46)

    BASIC BLOCK #1
          BEGINNING AT 0, ENDING AT 17
          BLOCK TRAVERSED 1 TIMES
          0 REFERENCES:
    BASIC BLOCK #2
          BEGINNING AT 17, ENDING AT 25
          BLOCK TRAVERSED 1 TIMES
          1 REFERENCES:
             1
    BASIC BLOCK #3
          BEGINNING AT 37, ENDING AT 45
          BLOCK TRAVERSED 1 TIMES
          1 REFERENCES:
             2
    BASIC BLOCK #4
          BEGINNING AT 26, ENDING AT 36
          BLOCK TRAVERSED 1 TIMES
          1 REFERENCES:
             2
BLOCK SUMMARY DATA:

BLOCK NUMBER                     NUMBER OF PASSES

     1                                  1
     2                                  1
     3                                  1
     4                                  1


TOTAL NUMBER OF BLOCKS PROCESSED WAS 4
USING THE STEEPEST DESCENT (MINIMUM CURRENT POOL)  BLOCK SELECTION ALGORITHM.

TIME FOR THE OPTIMIZATION WAS 0.233 SECONDS.

                   ***CODE SYNTHESIS FILTER IS COMPLETE***
```

III. TESTING OF BLOCK SELECTION METHOD

As previously mentioned, a principal objective of the implementation of Algorithm Q was to test block selection methods. Three classes of programs were analyzed using the CSF. A class of programs was developed by writing test programs that would encompass various typical program flows. This class is by far the more complex of the three test classes; however, it was felt that a "foreign" source for test programs should be sought to inject unbiased data. Using ALGOL-E programs written by students in an introductory ALGOL class, two samples were obtained. TABLES VI, VII and VIII are the synopses of the data compiled on these program sets. TABLES IX, X and XI give a statistical summary of the program classes while TABLE XII is a summary of all programs tested.

Lukasczyk [34] concluded that "steepest descent" outperformed LIFO, FIFO and the depth-first-search selection algorithm of Hecht and Ullman [23, 24]. The method used by Lukasczyk to heuristically arrive at steepest descent was to process the block with the minimum number of equivalence classes first. To observe the effects of increased complexity, the test programs were broken into five subclasses:

1) code size less than 200 words
2) code size from 200 to 400 words
3) code size from 400 to 600 words
4) code size from 600 to 800 words
5) code size greater than 800 words.

TABLES XIII, XIV, XV, XVI and XVII are the summary analyses of these subclasses.

Figure 9 is a graphical analysis of the five subclasses of program size tested. It shows that heuristic steepest descent results in a faster convergence rate for the global flow analysis algorithm given any of the code file sizes tested. The percentage of improvement between steepest descent and LIFO at each test level is:

LEVEL	PERCENTAGE IMPROVEMENT
less 200	12.5%
200 to 400	2.3%
400 to 600	4.4%
600 to 800	4.6%
greater 800	4.5%.

The improvement of steepest descent over LIFO is obvious and as Figure 9 shows, the other two methods tested are worse than LIFO in all cases. As can be seen, the percentage of improvement does not clearly justify the use of a specific selection method as program size increases. In fact, the supposition that the average number of blocks increases with code size is even suspect. Unfortunately, upon close examination the test data has defficiencies. The number of blocks processed for programs with a code file size from 0 to 400 words is biased from the generalized class of test programs. These programs were written to be as complex as possible and do not reflect a true "real world" program flow. The code file sizes analyzed are restricted to programs with less than 1000 words. This limitation resulted from the inability of the CSF to handle programs that include array structures. It is felt that the limited

sample size, coupled with the biased generalized programs has severely affected the test results.

TABLE VI

DATA FOR STUDENT TEST #1

NUMBER OPTIMS	CODE	BLOCKS	BLOCK SELECTION METHOD CHOSEN LIFO	FIFO	MIN POOL	MAX POOL
1	153	15	20	19	15	20
3	162	12	24	42	24	24
0	123	14	14	23	14	14
1	152	15	20	19	15	20
0	99	12	12	19	12	12
0	101	12	12	19	12	12
0	144	14	14	23	14	16
3	91	10	16	19	16	19
0	115	18	30	35	23	36
0	165	17	18	28	17	18
0	104	15	20	19	16	20
0	122	14	14	23	14	14
0	126	14	14	23	14	14
1	156	15	20	19	15	21
0	132	16	16	27	16	16
0	115	12	15	23	16	16
0	97	12	12	19	12	12
0	109	16	20	19	18	20
8	127	5	5	5	5	5
0	99	12	12	19	12	12
0	104	15	20	19	16	20
0	108	15	20	19	16	20
0	99	12	12	19	12	12
0	109	16	20	19	18	20
0	93	12	12	19	12	12
0	118	12	15	22	15	15

TABLE VI (CON'T)

0	96	12	12	19	12	12	
0	94	12	12	19	12	12	
0	101	12	12	19	12	12	
0	111	16	20	19	18	20	
0	71	12	12	19	12	12	
0	71	12	12	19	12	12	
0	81	16	23	21	19	23	
0	99	12	12	19	12	12	
0	93	12	12	19	12	12	
0	107	16	23	21	17	23	
0	107	16	23	21	17	23	
0	107	16	23	21	17	23	
0	87	16	20	19	17	20	
5	728	9	13	13	13	13	
0	102	15	20	19	16	20	

TABLE VII

DATA FOR STUDENT TEST #2

NUMBER OPTIMS	CODE	BLOCKS	BLOCK SELECTION METHOD CHOSEN LIFO	FIFO	MIN POOL	MAX POOL
2	161	24	53	51	38	68
4	513	9	13	13	13	13
5	540	9	14	14	14	14
5	728	9	13	13	13	13
6	719	11	18	18	18	18
9	859	12	19	22	19	20
4	544	9	13	13	13	13
4	979	9	13	13	13	13
7	348	31	66	70	74	76
6	665	11	15	15	14	15
4	812	13	18	21	16	16
7	708	10	16	20	15	15
5	625	9	16	16	16	16
5	582	10	14	14	14	14
4	928	11	15	15	15	15
6	781	14	20	22	19	20
5	693	12	16	16	16	16
11	307	25	42	48	44	46
9	286	18	40	43	38	42
10	296	24	42	56	43	64
6	418	16	26	30	24	26
8	584	17	28	33	26	28
11	926	19	31	38	29	34
10	629	21	36	43	32	38
10	289	25	43	56	45	48

TABLE VIII

GENERALIZED STRUCTURE TEST

NUMBER OPTIMS	CODE	BLOCKS	BLOCK SELECTION METHOD CHOSEN LIFO	FIFO	MIN POOL	MAX POOL
9	107	3	3	3	3	3
4	113	10	13	11	10	13
9	130	6	8	8	8	8
5	130	6	8	8	8	8
1	153	15	20	19	15	20
6	100	7	10	8	7	10
1	140	10	19	19	19	19
2	135	10	18	18	18	18
14	226	11	22	16	11	22
10	269	15	26	26	26	26
1	141	10	19	19	19	19
8	127	5	5	5	5	5
1	152	15	28	31	15	20
8	239	14	28	31	21	25
9	87	3	3	3	3	3
4	113	10	13	11	10	13
3	90	10	16	19	16	19
10	269	15	26	26	26	25
17	256	10	12	12	12	12
10	281	20	36	38	34	38
10	242	15	25	26	23	27
18	216	8	10	10	10	10
18	259	11	15	15	15	15
17	256	10	12	12	12	12
10	215	14	20	23	20	23
17	213	7	7	7	7	7
9	266	14	23	23	23	22
1	102	7	14	16	15	18

TABLE IX

SUMMARY FOR STUDENT TEST # 1

BLOCK SELECTION METHOD	AVG BLOCKS PROCESSED	AVG PASSES PER BLOCK
LIFO	16.9	1.2
FIFO	20.9	1.5
MIN POOL	15.1	1.1
MAX POOL	17.2	1.2

AVERAGE CODE GENERATED: 126

AVERAGE OPTIMIZATIONS DETECTED: .3

AVERAGE BLOCKS: 13.8

TABLE X

SUMMARY FOR STUDENT TEST # 2

BLOCK SELECTION METHOD	AVG BLOCKS PROCESSED	AVG PASSES PER BLOCK
LIFO	24.9	1.6
FIFO	27.4	1.8
MIN POOL	24.0	1.6
MAX POOL	27.2	1.7

AVERAGE CODE GENERATED: 610

AVERAGE OPTIMIZATIONS DETECTED: 6.4

AVERAGE BLOCKS: 14.7

TABLE XI

SUMMARY FOR GENERALIZED STRUCTURE TEST

BLOCK SELECTION METHOD	AVG BLOCKS PROCESSED	AVG PASSES PER BLOCK
LIFO	16.8	1.5
FIFO	16.9	1.5
MIN POOL	15.4	1.4
MAX POOL	16.4	1.5

AVERAGE CODE GENERATED: 185

AVERAGE OPTIMIZATIONS DETECTED: 8.3

AVERAGE BLOCKS: 10.8

TABLE XII

SUMMARY FOR ALL TESTS

BLOCK SELECTION METHOD	AVG BLOCKS PROCESSED	AVG PASSES PER BLOCK
LIFO	19.0	1.4
FIFO	21.5	1.6
MIN POOL	17.5	1.3
MAX POOL	19.6	1.4

AVERAGE CODE GENERATED: 270

AVERAGE OPTIMIZATIONS DETECTED: 4.2

AVERAGE BLOCKS: 13.2

TABLE XIII

ANALYSIS OF CODE SIZE LESS THAN 200 WORDS

BLOCK SELECTION METHOD	AVG BLOCKS PROCESSED	AVG PASSES PER BLOCK
LIFO	16.8	1.3
FIFO	19.8	1.5
MIN POOL	14.9	1.2
MAX POOL	17.4	1.3

AVERAGE CODE GENERATED: 115

AVERAGE OPTIMIZATIONS DETECTED: 1.1

AVERAGE BLOCKS: 12.9

TABLE XIV

ANALYSIS OF CODE SIZE FROM 200 TO 400 WORDS

BLOCK SELECTION METHOD	AVG BLOCKS PROCESSED	AVG PASSES PER BLOCK
LIFO	27.5	1.6
FIFO	29.9	1.7
MIN POOL	26.9	1.6
MAX POOL	28.9	1.6

AVERAGE CODE GENERATED: 263

AVERAGE OPTIMIZATIONS DETECTED: 11.9

AVERAGE BLOCKS: 15.9

TABLE XV

ANALYSIS OF CODE SIZE FROM 400 TO 600 WORDS

BLOCK SELECTION METHOD	AVG BLOCKS PROCESSED	AVG PASSES PER BLOCK
LIFO	18.9	1.5
FIFO	20.7	1.6
MIN POOL	18.1	1.5
MAX POOL	18.9	1.5

AVERAGE CODE GENERATED: 524

AVERAGE OPTIMIZATIONS DETECTED: 5.5

AVERAGE BLOCKS: 12.2

TABLE XVI

ANALYSIS OF CODE SIZE FROM 600 TO 800 WORDS

BLOCK SELECTION METHOD	AVG BLOCKS PROCESSED	AVG PASSES PER BLOCK
LIFO	17.8	1.5
FIFO	19.1	1.6
MIN POOL	17.0	1.5
MAX POOL	17.9	1.5

AVERAGE CODE GENERATED: 694

AVERAGE OPTIMIZATIONS DETECTED: 6.1

AVERAGE BLOCKS: 11.7

TABLE XVII

ANALYSIS OF CODE FILE SIZE GREATER THAN 800 WORDS

BLOCK SELECTION METHOD	AVG BLOCKS PROCESSED	AVG PASSES PER BLOCK
LIFO	19.2	1.5
FIFO	21.8	1.7
MIN POOL	18.4	1.4
MAX POOL	19.6	1.5

AVERAGE CODE GENERATED: 901

AVERAGE OPTIMIZATIONS DETECTED: 6.4

AVERAGE BLOCKS: 12.8

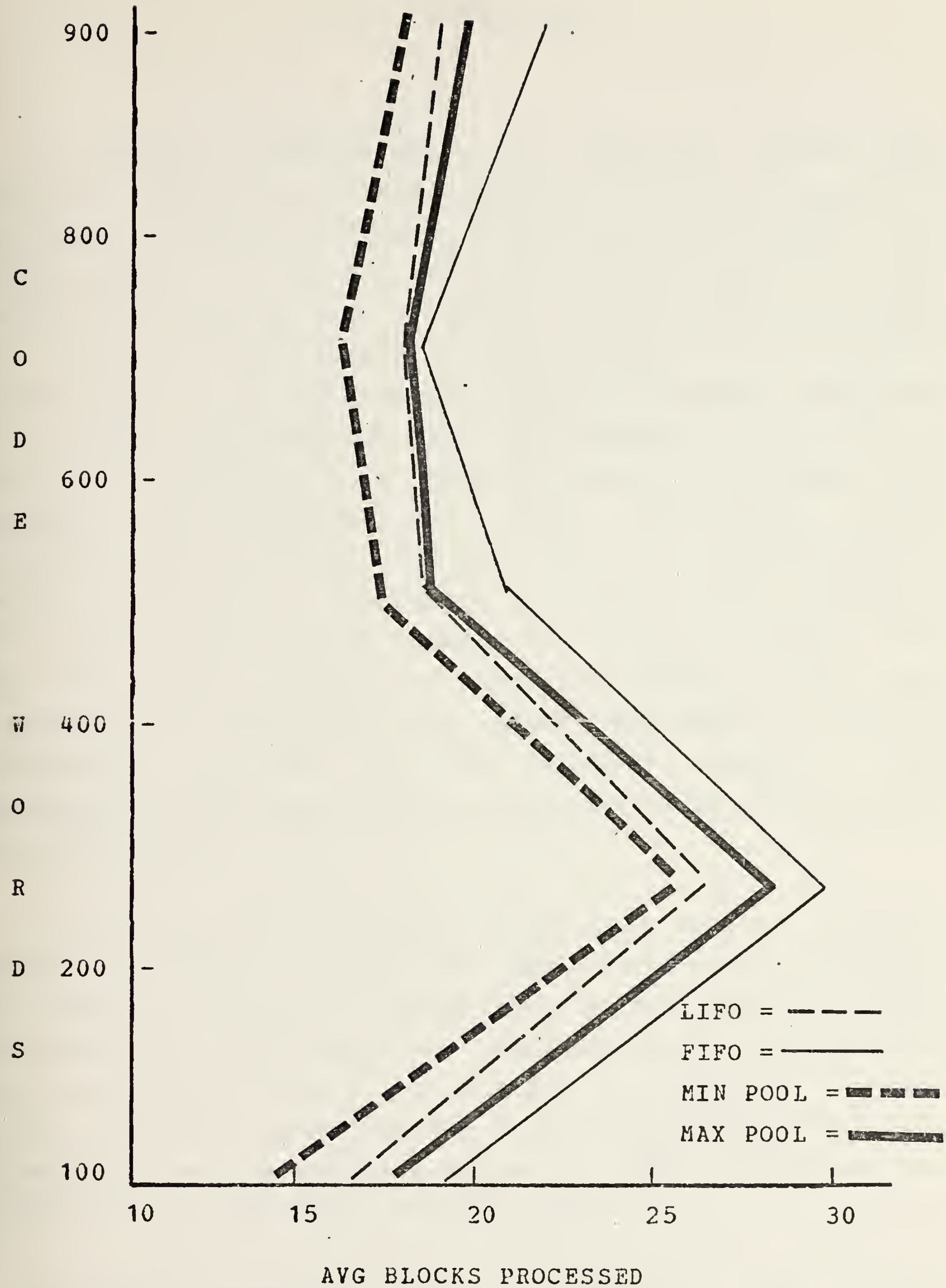

FIGURE 9
RELATIONSHIP OF CODE SIZE AND BLOCKS PROCESSED

IV. CONCLUSIONS

A complete implementation of Kildall's global flow analysis algorithm has been presented. However, testing of convergence rates by different block selection methods was inconclusive. It is felt that the primary reason for this was due to a lack of test programs. Unfortunately, programs are available but cannot be used because of current CSF limitations. The inability to handle subscripted variables is a primary limitation. If this capability were added to the CSF, not only would a greater amount of small test programs be available, but, more importantly, larger programs could be tested. Kildall [29] has implemented a method of handling subscripts in a similar CSF optimizer. This method assigns class numbers to an array element only when the exact element is known and assumes a worst case assumption by nullifying all array assignments when an assignment is made to an unknown element. The implementation of this method in the CSF would be relatively straightforward.

Of the four block selection algorithms tested, steepest descent was shown to result in a faster convergence rate for the global flow analysis algorithm in all test samples. Unfortunately, LIFO was the easiest to implement and its small performance loss in relation to steepest descent cannot really be justified due to the additional coding and longer run time for steepest descent. FIFO performed the poorest of all selection methods tested.

In summary, steepest descent resulted in the fastest convergence rate for the global flow analysis algorithm, but statistically the improvement shown is not significant. Except for the possible rejection of FIFO, no real

conclusions as to which block selection method is better can be substantiated.

Live expression analysis is an area of interest that was not approached in this version of the CSF. Kildall [28, 30] has shown that a backward pass through the program to detect live expressions would allow the use of the live expression pool at each node as the initial current pool instead of the identity operand for the defined meet operator. Hand analysis using this method show a marked improvement in the convergence rate of the algorithm. An effort to implement a live expression analysis into the CSF should be attempted since it would considerably improve the convergence rate of the algorithm.

COMPUTER OUTPUT

```
CARD |BL| SYL |
   1 | 0|   0 |COMMENT     TEST PROGRAM # 1        ;
$EXECUTE
   3 | 0|   1 |BEGIN LOCAL   A,B,C,D,E,F,G,H,I,J;
   4 | 1|   1 |         A := 10;
   5 | 1|   5 |         B := 3;
   6 | 1|   9 |       C := 20;
   7 | 1|  13 |       D := 5;
   8 | 1|  17 |       E := 20;
   9 | 1|  21 |       F := 10;
  10 | 1|  25 |       WHILE A GTR B DO
  11 | 1|  35 |              BEGIN
  12 | 1|  35 |              WHILE C GTR D DO
  13 | 2|  45 |                     BEGIN
  14 | 2|  45 |                     WHILE E GTR F DO
  15 | 3|  55 |                            BEGIN
  16 | 3|  55 |                            B := B + 1;
  17 | 4|  62 |                            F := F + 1;
  18 | 4|  69 |                            D := D + 1;
  19 | 4|  76 |                     END;
  20 | 3|  79 |                     B := B +1;
  21 | 3|  86 |                     D := D + 1;
  22 | 3|  93 |                     E := B;
  23 | 3|  98 |                     H := E * C;
  24.| 3| 106 |              END;
  25 | 2| 109 |              B := B + 1;
  26 | 2| 116 |              E := C;
  27 | 2| 121 |              J := C * B;
  28 | 2| 129 |              I := B * C;
  29 | 2| 137 |       END;
  30 | 1| 140 |END
  31 | 1| 140 |EOF
CODE FILE COPIED (140 WORDS)
CONSTANT TABLE COPIED (10 WORDS)
2 RECORDS WRITTEN INTO FILE 1
END OF COMPILATION FEBRUARY 9, 1975.  CLOCK TIME = 18:9:7.40.

31 CARDS WERE READ.
NO ERRORS WERE DETECTED.


SET UP TIME                  0:0:0.04.
ACTUAL COMPILATION TIME      0:0:0.80.
CLEAN-UP TIME AT END         0:0:0.01.
TOTAL TIME IN COMPILER       0:0:0.85.
COMPILATION RATE :2325 CARDS PER MINUTE.
```

```
                    ***CODE SYNTHESIS FILTER***

BLOCK SELECTION METHOD WILL BE STEEPEST DESCENT (MINIMUM CURRENT POOL)

***C O M P L E T E   T A B L E   D U M P***
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

***T A B L E   D U M P*** AT LOCATION = 0

CONSTANTS <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
CONSYM    |10  |3   |20  |5   |1
CONTYPE   |INT |INT |INT |INT |INT
CONVAL    |141 |142 |143 |144 |145
CONINT    |10  |3   |20  |5   |1

ADDR TABL <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

VALUE STACK (TOP AT 127)


CONT STK  <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
BLOCK#    |0
ENTRY     |0

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

 ****  FORMATTED INTERMEDIATE CODE DUMP  ****
 LOC OFF    OP CODE     CNS ADR     RAW CODE       OPTIMIZATION
     SET                    /ETC                        TYPE
*******************************************************************************
 0    0     |ENT        |C|141  -  | 0003808D  |
 1    0     |TOGGLE     | |257     | 00080101  |
 2    0     |LOC        | |1       | 00020001  |
 3    0     |INT        | |1       | 00050001  |
 4    0     |STO        | |0       | 00280000  |
 5    0     |DEL        | |0       | 002A0000  |
 6    0     |LOC        | |2       | 00020002  |
 7    0     |INT        | |2       | 00050002  |
 8    0     |STO        | |0       | 00280000  |
 9    0     |DEL        | |0       | 002A0000  |
 10   0     |LOC        | |3       | 00020003  |
 11   0     |INT        | |3       | 00050003  |
 12   0     |STO        | |0       | 00280000  |
 13   0     |DEL        | |0       | 002A0000  |
 14   0     |LOC        | |4       | 00020004  |
 15   0     |INT        | |4       | 00050004  |
 16   0     |STO        | |0       | 00280000  |
 17   0     |DEL        | |0       | 002A0000  |
 18   0     |LOC        | |5       | 00020005  |
 19   0     |INT        | |3       | 00050003  |
 20   0     |STO        | |0       | 00280000  |
 21   0     |DEL        | |0       | 002A0000  |
 22   0     |LOC        | |6       | 00020006  |
 23   0     |INT        | |1       | 00050001  |
 24   0     |STO        | |0       | 00280000  |
 25   0     |DEL        | |0       | 002A0000  |
 26   0     |ENT        | |1       | 00030001  |
 27   0     |LOC        | |1       | 00020001  |
 28   0     |LOD        | |0       | 00270000  |
 29   0     |LOC        | |2       | 00020002  |
 30   0     |LOD        | |0       | 00270000  |
 31   0     |GTR        | |0       | 00210000  |
 32   0     |XIT        | |140     | 0004008C  |
 33   0     |XIT        | |35      | 00040023  |
 34   0     |BSC        | |0       | 002F0000  |
 35   0     |ENT        | |1       | 00030001  |
 36   0     |ENT        | |1       | 00030001  |
 37   0     |LOC        | |3       | 00020003  |
 38   0     |LOD        | |0       | 00270000  |
 39   0     |LOC        | |4       | 00020004  |
 40   0     |LOD        | |0       | 00270000  |
 41   0     |GTR        | |0       | 00210000  |
```

42	0	XIT	109	0004006D
43	0	XIT	45	0004002D
44	0	BSC	0	002F0000
45	0	ENT	1	00030001
46	0	ENT	1	00030001
47	0	LOC	5	00020005
48	0	LOD	0	00270000
49	0	LOC	6	00020006
50	0	LOD	0	00270000
51	0	GTR	0	00210000
52	0	XIT	79	0004004F
53	0	XIT	55	00040037
54	0	BSC	0	002F0000
55	0	ENT	1	00030001
56	0	LOC	2	00020002
57	0	LOC	2	00020002
58	0	LOD	0	00270000
59	0	INT	5	00050005
60	0	ADD	0	00100000
61	0	STO	0	00280000
62	0	DEL	0	002A0000
63	0	LOC	6	00020006
64	0	LOC	6	00020006
65	0	LOD	0	00270000
66	0	INT	5	00050005
67	0	ADD	0	00100000
68	0	STO	0	00280000
69	0	DEL	0	002A0000
70	0	LOC	4	00020004
71	0	LOC	4	00020004
72	0	LOD	0	00270000
73	0	INT	5	00050005
74	0	ADD	0	00100000
75	0	STO	0	00280000
76	0	DEL	0	002A0000
77	0	XIT	46	0004002E
78	0	BRS	0	002E0000
79	0	ENT	1	00030001
80	0	LOC	2	00020002
81	0	LOC	2	00020002
82	0	LOD	0	00270000
83	0	INT	5	00050005
84	0	ADD	0	00100000
85	0	STO	0	00280000
86	0	DEL	0	002A0000
87	0	LOC	4	00020004
88	0	LOC	4	00020004
89	0	LOD	0	00270000
90	0	INT	5	00050005
91	0	ADD	0	00100000
92	0	STO	0	00280000
93	0	DEL	0	002A0000
94	0	LOC	5	00020005
95	0	LOC	2	00020002
96	0	LOD	0	00270000
97	0	STO	0	00280000
98	0	DEL	0	002A0000
99	0	LOC	8	00020008
100	0	LOC	5	00020005
101	0	LOD	0	00270000
102	0	LOC	3	00020003
103	0	LOD	0	00270000
104	0	MUL	0	00140000
105	0	STO	0	00280000
106	0	DEL	0	002A0000
107	0	XIT	36	00040024
108	0	BRS	0	002E0000
109	0	ENT	1	00030001
110	0	LOC	2	00020002
111	0	LOC	2	00020002
112	0	LOD	0	00270000
113	0	INT	5	00050005
114	0	ADD	0	00100000
115	0	STO	0	00280000
116	0	DEL	0	002A0000
117	0	LOC	5	00020005
118	0	LOC	3	00020003
119	0	LOD	0	00270000
120	0	STO	0	00280000
121	0	DEL	0	002A0000
122	0	LOC	10	0002000A
123	0	LOC	3	00020003
124	0	LOD	0	00270000
125	0	LOC	2	00020002
126	0	LOD	0	00270000
127	0	MUL	0	00140000
128	0	STO	0	00280000
129	0	DEL	0	002A0000

```
130    0    |LOC        |  |9        | 00020009   |
131    0    |LOC        |  |2        | 00020002   |
132    0    |LOD        |  |0        | 00270000   |
133    0    |LOC        |  |3        | 00020003   |
134    0    |LOD        |  |0        | 00270000   |
135    0    |MUL        |  |0        | 00140000   |
136    0    |STO        |  |0        | 00280000   |
137    0    |DEL        |  |0        | 002A0000   |
138    0    |XIT        |  |26       | 0004001A   |
139    0    |BRS        |  |0        | 002E0000   |
140    0    |ENT        |  |1        | 00030001   |
************************************************************************************************

        BASIC BLOCK #1
                BEGINNING AT 0, ENDING AT 0
                BLOCK TRAVERSED 0 TIMES
                0 REFERENCES:

***E N D    O F    C O M P L E T E    T A B L E    D U M P***

B+2    0    |LOC        |  |1        | 00020001   |
B+3    0    |INT        |  |1        | 00050001   |
B+4    0    |STO        |  |0        | 00280000   |
B+5    0    |DEL        |  |0        | 002A0000   |
B+6    0    |LOC        |  |2        | 00020002   |
B+7    0    |INT        |  |2        | 00050002   |
B+8    0    |STO        |  |0        | 00280000   |
B+9    0    |DEL        |  |0        | 002A0000   |
B+10   0    |LOC        |  |3        | 00020003   |
B+11   0    |INT        |  |3        | 00050003   |
B+12   0    |STO        |  |0        | 00280000   |
B+13   0    |DEL        |  |0        | 002A0000   |
B+14   0    |LOC        |  |4        | 00020004   |
B+15   0    |INT        |  |4        | 00050004   |
B+16   0    |STO        |  |0        | 00280000   |
B+17   0    |DEL        |  |0        | 002A0000   |
B+18   0    |LOC        |  |5        | 00020005   |
B+19   0    |INT        |  |3        | 00050003   |
B+20   0    |STO        |  |0        | 00280000   |
B+21   0    |DEL        |  |0        | 002A0000   |
B+22   0    |LOC        |  |6        | 00020006   |
B+23   0    |INT        |  |1        | 00050001   |
B+24   0    |STO        |  |0        | 00280000   |
B+25   0    |DEL        |  |0        | 002A0000   |
B+26   0    |ENT        |  |1        | 00030001   |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 2 IS:
 A(6,141,141)  A(5,143,143)  A(4,144,144)  A(3,143,143)  A(2,142,142)
 A(1,141,141)

 CONTROL AT 26, FROM BLK 1 TO BLK 2 (PASS 1)
C+26   0    |ENT        |C|145      | 00038091   |
B+27   0    |LOC        |  |1        | 00020001   |
B+28   0    |LOD        |  |0        | 00270000   |
B+29   0    |LOC        |  |2        | 00020002   |
B+30   0    |LOD        |  |0        | 00270000   |
B+31   0    |GTR        |  |0        | 00210000   |
B+32   0    |XIT        |  |140      | 0004008C   |
B+33   0    |XIT        |  |35       | 00040023   |
B+34   0    |BSC        |  |0        | 002F0000   |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 3 IS:
 C(141:142,*,147)  A(6,141,141)  A(5,143,143)  A(4,144,144)  A(3,143,143)
 A(2,142,142)  A(1,141,141)

 CONTROL AT 35, FROM BLK 2 TO BLK 3 (PASS 1)
C+35   0    |ENT        |C|150      | 00038096   |
B+36   0    |ENT        |  |1        | 00030001   |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 4 IS:
 C(141:142,*,147)  A(6,141,141)  A(5,143,143)  A(4,144,144)  A(3,143,143)
 A(2,142,142)  A(1,141,141)

 CONTROL AT 36, FROM BLK 3 TO BLK 4 (PASS 1)
C+36   0    |ENT        |C|155      | 0003809B   |
B+37   0    |LOC        |  |3        | 00020003   |
B+38   0    |LOD        |  |0        | 00270000   |
B+39   0    |LOC        |  |4        | 00020004   |
B+40   0    |LOD        |  |0        | 00270000   |
B+41   0    |GTR        |  |0        | 00210000   |
B+42   0    |XIT        |  |109      | 0004006D   |
B+43   0    |XIT        |  |45       | 0004002D   |
B+44   0    |BSC        |  |0        | 002F0000   |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 5 IS:
 C(141:142,*,147)  C(143:144,*,147)  A(6,141,141)  A(5,143,143)  A(4,144,144)
 A(3,143,143)  A(2,142,142)  A(1,141,141)
```

```
CONTROL AT 45, FROM BLK 4 TO BLK 5 (PASS 1)
C+45    0   |ENT            |C|160   | 000380A0  |
B+46    0   |ENT            | |1     | 00030001  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 6 IS:
 C(143:144,*,147)  C(141:142,*,147)  A(6,141,141)  A(5,143,143)  A(4,144,144)
 A(3,143,143)  A(2,142,142)  A(1,141,141)

 CONTROL AT 46, FROM BLK 5 TO BLK 6 (PASS 1)
C+46    0   |ENT            |C|165   | 000380A5  |
B+47    0   |LOC            | |5     | 00020005  |
B+48    0   |LOD            | |0     | 00270000  |
B+49    0   |LOC            | |6     | 00020006  |
B+50    0   |LOD            | |0     | 00270000  |
B+51    0   |GTR            | |0     | 00210000  |
B+52    0   |XIT            | |79    | 0004004F  |
B+53    0   |XIT            | |55    | 00040037  |
B+54    0   |BSC            | |0     | 002F0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 7 IS:
 C(141:142,*,147)  C(143:144,*,147)  C(143:141,*,147)  A(6,141,141)
 A(5,143,143)  A(4,144,144)  A(3,143,143)  A(2,142,142)  A(1,141,141)

 CONTROL AT 55, FROM BLK 6 TO BLK 7 (PASS 1)
C+55    0   |ENT            |C|170   | 000380AA  |
B+56    0   |LOC            | |2     | 00020002  |
B+57    0   |LOC            | |2     | 00020002  |
B+58    0   |LOD            | |0     | 00270000  |
B+59    0   |INT            | |5     | 00050005  |
B+60    0   |ADD            | |0     | 00100000  |
B+61    0   |STO            | |0     | 00280000  |
B+62    0   |DEL            | |0     | 002A0000  |
B+63    0   |LOC            | |6     | 00020006  |
B+64    0   |LOC            | |6     | 00020006  |
B+65    0   |LOD            | |0     | 00270000  |
B+66    0   |INT            | |5     | 00050005  |
B+67    0   |ADD            | |0     | 00100000  |
B+68    0   |STO            | |0     | 00280000  |
B+69    0   |DEL            | |0     | 002A0000  |
B+70    0   |LOC            | |4     | 00020004  |
B+71    0   |LOC            | |4     | 00020004  |
B+72    0   |LOD            | |0     | 00270000  |
B+73    0   |INT            | |5     | 00050005  |
B+74    0   |ADD            | |0     | 00100000  |
B+75    0   |STO            | |0     | 00280000  |
B+76    0   |DEL            | |0     | 002A0000  |
B+77    0   |XIT            | |46    | 0004002E  |
B+78    0   |BRS            | |0     | 002E0000  |

BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 7 IS:
 C(143:141,*,147)  C(143:144,*,147)  C(141:142,*,147)  C(142:145,*,151)
 C(141:145,*,153)  C(144:145,*,155)  A(4,155,155)  A(6,153,153)  A(2,151,151)
 A(5,143,143)  A(3,143,143)  A(1,141,141)

 CURRENT POOL FOR BLOCK# 6 IS:
 C(143:144,*,147)  C(141:142,*,147)  A(6,141,141)  A(5,143,143)  A(4,144,144)
 A(3,143,143)  A(2,142,142)  A(1,141,141)

AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 6 IS:
 A(6,*,156)  A(5,143,143)  A(4,*,157)  A(3,143,143)  A(2,*,158)  A(1,141,141)

 CONTROL AT 46, FROM BLK 7 TO BLK 6 (PASS 2)
C+46    0   |ENT            |C|165   | 000380A5  |
B+47    0   |LOC            | |5     | 00020005  |
B+48    0   |LOD            | |0     | 00270000  |
B+49    0   |LOC            | |6     | 00020006  |
B+50    0   |LOD            | |0     | 00270000  |
B+51    0   |GTR            |C|1004  | 002183EC  |   CONSTANT PROPAGATION
B+52    0   |XIT            | |79    | 0004004F  |
B+53    0   |XIT            | |55    | 00040037  |
B+54    0   |BSC            | |0     | 002F0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 8 IS:
 C(143:156,*,159)  A(6,*,156)  A(5,143,143)  A(4,*,157)  A(3,143,143)
 A(2,*,158)  A(1,141,141)

BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 6 IS:
 C(143:156,*,159)  A(6,*,156)  A(5,143,143)  A(4,*,157)  A(3,143,143)
 A(2,*,158)  A(1,141,141)
```

```
CURRENT POOL FOR BLOCK# 7 IS:
C(141:142,*,147)  C(143:144,*,147)  C(143:141,*,147)  A(6,141,141)
A(5,143,143)  A(4,144,144)  A(3,143,143)  A(2,142,142)  A(1,141,141)

AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 7 IS:
 A(6,*,160)  A(5,143,143)  A(4,*,161)  A(3,143,143)  A(2,*,162)  A(1,141,141)

 CONTROL AT 55, FROM BLK 6 TO BLK 7 (PASS 2)
C+55    0     |ENT          |C|170     |  00038DAA  |
B+56    0     |LOC          | |2       |  00020002  |
B+57    0     |LOC          | |2       |  00020002  |
B+58    0     |LOD          | |0       |  00270000  |
B+59    0     |INT          | |5       |  00050005  |
B+60    0     |ADD          |C|994     |  001083E2  |   CONSTANT PROPAGATION
B+61    0     |STO          | |0       |  00280000  |
B+62    0     |DEL          | |0       |  002A0000  |
B+63    0     |LOC          | |6       |  00020006  |
B+64    0     |LOC          | |6       |  00020006  |
B+65    0     |LOD          | |0       |  00270000  |
B+66    0     |INT          | |5       |  00050005  |
B+67    0     |ADD          |C|990     |  001083DE  |   CONSTANT PROPAGATION
B+68    0     |STO          | |0       |  00280000  |
B+69    0     |DEL          | |0       |  002A0000  |
B+70    0     |LOC          | |4       |  00020004  |
B+71    0     |LOC          | |4       |  00020004  |
B+72    0     |LOD          | |0       |  00270000  |
B+73    0     |INT          | |5       |  00050005  |
B+74    0     |ADD          |C|986     |  001083DA  |   CONSTANT PROPAGATION
B+75    0     |STO          | |0       |  00280000  |
B+76    0     |DEL          | |0       |  002A0000  |
B+77    0     |XIT          | |46      |  0004002E  |
B+78    0     |BRS          | |0       |  002E0000  |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 7 IS:
 C(162:145,*,163)  C(160:145,*,164)  C(161:145,*,165)  A(4,*,155)  A(6,*,164)
 A(2,*,163)  A(5,143,143)  A(3,143,143)  A(1,141,141)


 CURRENT POOL FOR BLOCK# 6 IS:
 A(6,*,156)  A(5,143,143)  A(4,*,157)  A(3,143,143)  A(2,*,158)  A(1,141,141)

AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

 CONTROL AT 79, FROM BLK 6 TO BLK 8 (PASS 1)
C+79    0     |ENT          |C|175     |  000380AF  |
B+80    0     |LOC          | |2       |  00020002  |
B+81    0     |LOC          | |2       |  00020002  |
B+82    0     |LOD          | |0       |  00270000  |
B+83    0     |INT          | |5       |  00050005  |
B+84    0     |ADD          | |0       |  00100000  |
B+85    0     |STO          | |0       |  00280000  |
B+86    0     |DEL          | |0       |  002A0000  |
B+87    0     |LOC          | |4       |  00020004  |
B+88    0     |LOC          | |4       |  00020004  |
B+89    0     |LOD          | |0       |  00270000  |
B+90    0     |INT          | |5       |  00050005  |
B+91    0     |ADD          | |0       |  00100000  |
B+92    0     |STO          | |0       |  00280000  |
B+93    0     |DEL          | |0       |  002A0000  |
B+94    0     |LOC          | |5       |  00020005  |
B+95    0     |LOC          | |2       |  00020002  |
B+96    0     |LOD          | |0       |  00270000  |
B+97    0     |STO          | |0       |  00280000  |
B+98    0     |DEL          | |0       |  002A0000  |
B+99    0     |LOC          | |8       |  00020008  |
B+100   0     |LOC          | |5       |  00020005  |
B+101   0     |LOD          | |0       |  00270000  |
B+102   0     |LOC          | |3       |  00020003  |
B+103   0     |LOD          | |0       |  00270000  |
B+104   0     |MUL          | |0       |  00140000  |
B+105   0     |STO          | |0       |  00280000  |
B+106   0     |DEL          | |0       |  002A0000  |
B+107   0     |XIT          | |36      |  00040024  |
B+108   0     |BRS          | |0       |  002E0000  |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 8 IS:
 C(143:156,*,159)  C(158:145,*,169)  C(157:145,*,170)  C(169:143,*,171)
 C(143:156,*,171)  A(8,*,171)  A(5,*,169)  A(4,*,170)  A(2,*,169)  A(6,*,156)
 A(3,143,143)  A(1,141,141)
```

```
 CURRENT POOL FOR BLOCK# 4 IS:
 C(141:142,*,147)  A(6,141,141)  A(5,143,143)  A(4,144,144)  A(3,143,143)
 A(2,142,142)  A(1,141,141)

AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 4 IS:
 A(6,*,172)  A(5,*,173)  A(4,*,174)  A(3,143,143)  A(2,*,175)  A(1,141,141)


 CONTROL AT 36, FROM BLK 8 TO BLK 4 (PASS 2)
C+36   0   |ENT        |C|155    | 0003809B |
B+37   0   |LOC        | |3      | 00020003 |
B+38   0   |LOD        | |0      | 00270000 |
B+39   0   |LOC        | |4      | 00020004 |
B+40   0   |LOD        | |0      | 00270000 |
B+41   0   |GTR        |C|1016   | 002183F8 |   CONSTANT PROPAGATION
B+42   0   |XIT        | |109    | 0004006D |
B+43   0   |XIT        | |45     | 0004002D |
B+44   0   |BSC        | |0      | 002F0000 |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 9 IS:
 C(143:174,*,176)  A(6,*,172)  A(5,*,173)  A(4,*,174)  A(3,143,143)
 A(2,*,175)  A(1,141,141)


BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 4 IS:
 C(143:174,*,176)  A(6,*,172)  A(5,*,173)  A(4,*,174)  A(3,143,143)
 A(2,*,175)  A(1,141,141)


 CURRENT POOL FOR BLOCK# 5 IS:
 C(141:142,*,147)  C(143:144,*,147)  A(6,141,141)  A(5,143,143)  A(4,144,144)
 A(3,143,143)  A(2,142,142)  A(1,141,141)
AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 5 IS:
 A(6,*,177)  A(5,*,178)  A(4,*,179)  A(3,143,143)  A(2,*,180)  A(1,141,141)


 CONTROL AT 45, FROM BLK 4 TO BLK 5 (PASS 2)
C+45   0   |ENT        |C|160    | 000380A0 |
B+46   0   |ENT        |C|165    | 000380A5 |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 5 IS:
 A(6,*,177)  A(5,*,178)  A(4,*,179)  A(3,143,143)  A(2,*,180)  A(1,141,141)


 CURRENT POOL FOR BLOCK# 6 IS:
 A(6,*,166)  A(5,143,143)  A(4,*,167)  A(3,143,143)  A(2,*,168)  A(1,141,141)
AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 6 IS:
 A(6,*,181)  A(5,*,182)  A(4,*,183)  A(3,143,143)  A(2,*,184)  A(1,141,141)


 CONTROL AT 46, FROM BLK 5 TO BLK 6 (PASS 3)
C+46   0   |ENT        |C|165    | 000380A5 |
B+47   0   |LOC        | |5      | 00020005 |
B+48   0   |LOD        | |0      | 00270000 |
B+49   0   |LOC        | |6      | 00020006 |
B+50   0   |LOD        | |0      | 00270000 |
B+51   0   |GTR        |C|1004   | 002183EC |
B+52   0   |XIT        | |79     | 0004004F |
B+53   0   |XIT        | |55     | 00040037 |
B+54   0   |BSC        | |0      | 002F0000 |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 6 IS:
 C(182:181,*,185)  A(6,*,181)  A(5,*,182)  A(4,*,183)  A(3,143,143)
 A(2,*,184)  A(1,141,141)


 CURRENT POOL FOR BLOCK# 8 IS:
 C(143:156,*,159)  A(6,*,156)  A(5,143,143)  A(4,*,157)  A(3,143,143)
 A(2,*,158)  A(1,141,141)
```

```
AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOC<# 8 IS:
 A(6,*,186)  A(5,*,187)  A(4,*,188)  A(3,143,143)  A(2,*,189)  A(1,141,141)


BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 6 IS:
 C(182:181,*,185)  A(6,*,181)  A(5,*,182)  A(4,*,183)  A(3,143,143)
 A(2,*,184)  A(1,141,141)


 CURRENT POOL FOR BLOCK# 7 IS:
 A(6,*,160)  A(5,143,143)  A(4,*,161)  A(3,143,143)  A(2,*,162)  A(1,141,141)
AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 7 IS:
 A(6,*,190)  A(5,*,191)  A(4,*,192)  A(3,143,143)  A(2,*,193)  A(1,141,141)


 CONTROL AT 55, FROM BLK 6 TO BLK 7 (PASS 3)
C+55    0   |ENT        |C|170     | 000380AA  |
B+56    0   |LOC        | |2       | 00020002  |
B+57    0   |LOC        | |2       | 00020002  |
B+58    0   |LOD        | |0       | 00270000  |
B+59    0   |INT        | |5       | 00050005  |
B+60    0   |ADD        |C|994     | 001083E2  |
B+61    0   |STO        | |0       | 00280000  |
B+62    0   |DEL        | |0       | 002A0000  |
B+63    0   |LOC        | |6       | 00020006  |
B+64    0   |LOC        | |6       | 00020006  |
B+65    0   |LOD        | |0       | 00270000  |
B+66    0   |INT        | |5       | 00050005  |
B+67    0   |ADD        |C|990     | 0010830E  |
B+68    0   |STO        | |0       | 00280000  |
B+69    0   |DEL        | |0       | 002A0000  |
B+70    0   |LOC        | |4       | 00020004  |
B+71    0   |LOC        | |4       | 00020004  |
B+72    0   |LOD        | |0       | 00270000  |
B+73    0   |INT        | |5       | 00050005  |
B+74    0   |ADD        |C|986     | 0010830A  |
B+75    0   |STO        | |0       | 00280000  |
B+76    0   |DEL        | |0       | 002A0000  |
B+77    0   |XIT        | |46      | 0004002E  |
B+78    0   |BRS        | |0       | 002E0000  |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 7 IS:
 C(193:145,*,194)  C(190:145,*,195)  C(192:145,*,196)  A(4,*,196)  A(6,*,195)
 A(2,*,194)  A(5,*,191)  A(3,143,143)  A(1,141,141)


 CURRENT POOL FOR BLOCK# 6 IS:
 A(6,*,181)  A(5,*,182)  A(4,*,183)  A(3,143,143)  A(2,*,184)  A(1,141,141)

AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

 CONTROL AT 79, FROM BLK 6 TO BLK 8 (PASS 2)
C+79    0   |ENT        |C|175     | 000380AF  |
B+80    0   |LOC        | |2       | 00020002  |
B+81    0   |LOC        | |2       | 00020002  |
B+82    0   |LOD        | |0       | 00270000  |
B+83    0   |INT        | |5       | 00050005  |
B+84    0   |ADD        |C|980     | 001083D4  |
B+85    0   |STO        | |0       | 00280000  |
B+86    0   |DEL        | |0       | 002A0000  |
B+87    0   |LOC        | |4       | 00020004  |
B+88    0   |LOC        | |4       | 00020004  |
B+89    0   |LOD        | |0       | 00270000  |
B+90    0   |INT        | |5       | 00050005  |
B+91    0   |ADD        |C|976     | 001083D0  |
B+92    0   |STO        | |0       | 00280000  |
B+93    0   |DEL        | |0       | 002A0000  |
B+94    0   |LOC        | |5       | 00020005  |
B+95    0   |LOC        | |2       | 00020002  |
B+96    0   |LOD        | |0       | 00270000  |
B+97    0   |STO        | |0       | 00280000  |
B+98    0   |DEL        | |0       | 002A0000  |
B+99    0   |LOC        | |8       | 00020008  |
B+100   0   |LOC        | |5       | 00020005  |
B+101   0   |LOD        | |0       | 00270000  |
B+102   0   |LOC        | |3       | 00020003  |
B+103   0   |LOD        | |0       | 00270000  |
```

```
B+104  0   |MUL            |C|972       | 001483CC  |
B+105  0   |STO            | |0         | 00280000  |
B+106  0   |DEL            | |0         | 002A0000  |
B+107  0   |XIT            | |36        | 00040024  |
B+108  0   |BRS            | |0         | 002E0000  |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 8 IS:
 C(189:145,*,201)  C(188:145,*,202)  C(201:143,*,203)  A(8,*,203)  A(5,*,201)
 A(4,*,202)  A(2,*,201)  A(6,*,186)  A(3,143,143)  A(1,141,141)

 CURRENT POOL FOR BLOCK# 4 IS:
 A(6,*,172)  A(5,*,173)  A(4,*,174)  A(3,143,143)  A(2,*,175)  A(1,141,141)
AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

 CONTROL AT 109, FROM BLK 4 TO BLK 9 (PASS 1)
C+109  0   |ENT            |C|180       | 000380B4  |
B+110  0   |LOC            | |2         | 00020002  |
B+111  0   |LOC            | |2         | 00020002  |
B+112  0   |LOD            | |0         | 00270000  |
B+113  0   |INT            | |5         | 00050005  |
B+114  0   |ADD            | |0         | 00100000  |
B+115  0   |STO            | |0         | 00280000  |
B+116  0   |DEL            | |0         | 002A0000  |
B+117  0   |LOC            | |5         | 00020005  |
B+118  0   |LOC            | |3         | 00020003  |
B+119  0   |LOD            | |0         | 00270000  |
B+120  0   |STO            | |0         | 00280000  |
B+121  0   |DEL            | |0         | 002A0000  |
B+122  0   |LOC            | |10        | 0002000A  |
B+123  0   |LOC            | |3         | 00020003  |
B+124  0   |LOD            | |0         | 00270000  |
B+125  0   |LOC            | |2         | 00020002  |
B+126  0   |LOD            | |0         | 00270000  |
B+127  0   |MUL            | |0         | 00140000  |
B+128  0   |STO            | |0         | 00280000  |
B+129  0   |DEL            | |0         | 002A0000  |
B+130  0   |LOC            | |9         | 00020009  |
B+131  0   |LOC            | |2         | 00020002  |
B+132  0   |LOD            | |0         | 00270000  |
B+133  0   |LOC            | |3         | 00020003  |
B+134  0   |LOD            | |0         | 00270000  |
B+135  0   |MUL            | |0         | 00140000  |
B+136  0   |STO            | |0         | 00280000  |
B+137  0   |DEL            | |0         | 002A0000  |
B+138  0   |XIT            | |26        | 0004001A  |
B+139  0   |BRS            | |0         | 002E0000  |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 9 IS:
 C(143:174,*,176)  C(175:145,*,208)  C(143:208,*,209)  A(9,*,209)  A(10,*,209)
 A(5,143,143)  A(2,*,208)  A(6,*,172)  A(4,*,174)  A(3,143,143)  A(1,141,141)

 CURRENT POOL FOR BLOCK# 2 IS:
 A(6,141,141)  A(5,143,143)  A(4,144,144)  A(3,143,143)  A(2,142,142)
 A(1,141,141)
AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 2 IS:
 A(6,*,210)  A(5,143,143)  A(4,*,211)  A(3,143,143)  A(2,*,212)  A(1,141,141)

 CONTROL AT 26, FROM BLK 9 TO BLK 2 (PASS 2)
C+26   0   |ENT            |C|145       | 00038091  |
B+27   0   |LOC            | |1         | 00020001  |
B+28   0   |LOD            | |0         | 00270000  |
B+29   0   |LOC            | |2         | 00020002  |
B+30   0   |LOD            | |0         | 00270000  |
B+31   0   |GTR            |C|1024      | 00218400  |   CONSTANT PROPAGATION
B+32   0   |XIT            | |140       | 0004008C  |
B+33   0   |XIT            | |35        | 00040023  |
B+34   0   |BSC            | |0         | 002F0000  |
 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 10 IS:
 C(141:212,*,213)  A(6,*,210)  A(5,143,143)  A(4,*,211)  A(3,143,143)
 A(2,*,212)  A(1,141,141)

BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 2 IS:
 C(141:212,*,213)  A(6,*,210)  A(5,143,143)  A(4,*,211)  A(3,143,143)
 A(2,*,212)  A(1,141,141)
```

```
CURRENT POOL FOR BLOCK# 3 IS:
C(141:142,*,147)  A(6,141,141)   A(5,143,143)   A(4,144,144)   A(3,143,143)
A(2,142,142)   A(1,141,141)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR  BLOC<# 3 IS:
A(6,*,214)   A(5,143,143)   A(4,*,215)   A(3,143,143)   A(2,*,216)   A(1,141,141)

CONTROL AT 35, FROM BLK 2 TO BLK 3 (PASS 2)
C+35    0    |ENT           |C|150     | 00038096    |
B+36    0    |ENT           |C|155     | 0003809B    |

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 3 IS:
A(6,*,214)   A(5,143,143)   A(4,*,215)   A(3,143,143)   A(2,*,216)   A(1,141,141)

CURRENT POOL FOR BLOCK# 4 IS:
A(6,*,204)   A(5,*,205)   A(4,*,206)   A(3,143,143)   A(2,*,207)   A(1,141,141)

AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

CONTROL AT 140, FROM BLK 2 TO BLK 10 (PASS 1)
C+140   0    |ENT           |C|185     | 000380B9    |

                                FINAL OPTIMIZATION RESULTS
#####################################################################################
-------------------------------------------------------------------------------------

OPTIMIZATION AT 135
        (143,208|127,209)    COMM SUBEXP ELIM

 ****  FORMATTED INTERMEDIATE CODE DUMP  ****
 LOC OFF     OP CODE     CNS ADR     RAW CODE        OPTIMIZATION
     SET                     /ETC                        TYPE
**************************************************************************
 0      0    |ENT         |  |141      | 0003008D    |
 1      0    |TOGGLE      |  |257      | 00080101    |
 2      0    |LOC         |  |1        | 00020001    |
 3      0    |INT         |  |1        | 00050001    |
 4      0    |STO         |  |0        | 00280000    |
 5      0    |DEL         |  |0        | 002A0000    |
 6      0    |LOC         |  |2        | 00020002    |
 7      0    |INT         |  |2        | 00050002    |
 8      0    |STO         |  |0        | 00280000    |
 9      0    |DEL         |  |0        | 002A0000    |
 10     0    |LOC         |  |3        | 00020003    |
 11     0    |INT         |  |3        | 00050003    |
 12     0    |STO         |  |0        | 00280000    |
 13     0    |DEL         |  |0        | 002A0000    |
 14     0    |LOC         |  |4        | 00020004    |
 15     0    |INT         |  |4        | 00050004    |
 16     0    |STO         |  |0        | 00280000    |
 17     0    |DEL         |  |0        | 002A0000    |
 18     0    |LOC         |  |5        | 00020005    |
 19     0    |INT         |  |3        | 00050003    |
 20     0    |STO         |  |0        | 00280000    |
 21     0    |DEL         |  |0        | 002A0000    |
 22     0    |LOC         |  |6        | 00020006    |
 23     0    |INT         |  |1        | 00050001    |
 24     0    |STO         |  |0        | 00280000    |
 25     0    |DEL         |  |0        | 002A0000    |
 26     0    |ENT         |  |145      | 00030091    |
 27     0    |LOC         |  |1        | 00020001    |
 28     0    |LOD         |  |0        | 00270000    |
 29     0    |LOC         |  |2        | 00020002    |
 30     0    |LOD         |  |0        | 00270000    |
 31     0    |GTR         |  |1024     | 00210400    |
 32     0    |XIT         |  |140      | 0004008C    |
 33     0    |XIT         |  |35       | 00040023    |
 34     0    |BSC         |  |0        | 002F0000    |
 35     0    |ENT         |  |150      | 00030096    |
 36     0    |ENT         |  |155      | 0003009B    |
 37     0    |LOC         |  |3        | 00020003    |
 38     0    |LOD         |  |0        | 00270000    |
 39     0    |LOC         |  |4        | 00020004    |
 40     0    |LOD         |  |0        | 00270000    |
```

41	0	GTR	1016	002103F8
42	0	XIT	109	0004006D
43	0	XIT	45	00040020
44	0	BSC	0	002F0000
45	0	ENT	160	000300A0
46	0	ENT	165	000300A5
47	0	LOC	5	00020005
48	0	LOD	0	00270000
49	0	LOC	6	00020006
50	0	LOD	0	00270000
51	0	GTR	1004	002103EC
52	0	XIT	79	0004004F
53	0	XIT	55	00040037
54	0	BSC	0	002F0000
55	0	ENT	170	000300AA
56	0	LOC	2	00020002
57	0	LOC	2	00020002
58	0	LOD	0	00270000
59	0	INT	5	00050005
60	0	ADD	994	001003E2
61	0	STO	0	00280000
62	0	DEL	0	002A0000
63	0	LOC	6	00020006
64	0	LOC	6	00020006
65	0	LOD	0	00270000
66	0	INT	5	00050005
67	0	ADD	990	001003DE
68	0	STO	0	00280000
69	0	DEL	0	002A0000
70	0	LOC	4	00020004
71	0	LOC	4	00020004
72	0	LOD	0	00270000
73	0	INT	5	00050005
74	0	ADO	986	001003DA
75	0	STO	0	00280000
76	0	DEL	0	002A0000
77	0	XIT	46	0004002E
78	0	BRS	0	002E0000
79	0	ENT	175	000300AF
80	0	LOC	2	00020002
81	0	LOC	2	00020002
82	0	LOD	0	00270000
83	0	INT	5	00050005
84	0	ADD	980	001003D4
85	0	STO	0	00280000
86	0	DEL	0	002A0000
87	0	LOC	4	00020004
88	0	LOC	4	00020004
89	0	LOD	0	00270000
90	0	INT	5	00050005
91	0	ADD	976	001003D0
92	0	STO	0	00280000
93	0	DEL	0	002A0000
94	0	LOC	5	00020005
95	0	LOC	2	00020002
96	0	LOO	0	00270000
97	0	STO	0	00280000
98	0	DEL	0	002A0000
99	0	LOC	8	00020008
100	0	LOC	5	00020005
101	0	LOD	0	00270000
102	0	LOC	3	00020003
103	0	LOD	0	00270000
104	0	MUL	972	001403CC
105	0	STO	0	00280000
106	0	DEL	0	002A0000
107	0	XIT	36	00040024
108	0	BRS	0	002E0000
109	0	ENT	180	000300B4
110	0	LOC	2	00020002
111	0	LOC	2	00020002
112	0	LOD	0	00270000
113	0	INT	5	00050005
114	0	AOD	966	001003C6
115	0	STO	0	00280000
116	0	DEL	0	002A0000
117	0	LOC	5	00020005
118	0	LOC	3	00020003
119	0	LOD	0	00270000
120	0	STO	0	00280000
121	0	DEL	0	002A0000
122	0	LOC	10	0002000A
123	0	LOC	3	00020003
124	0	LOD	0	00270000
125	0	LOC	2	00020002
126	0	LOD	0	00270000
127	0	MUL	962	001403C2
128	0	STO	0	00280000

```
129   0   |DEL        |  |0      | 002A0000 |
130   0   |LOC        |  |9      | 00020009 |
131   0   |LOC        |  |2      | 00020002 |
132   0   |LOD        |  |0      | 00270000 |
133   0   |LOC        |  |3      | 00020003 |
134   0   |LOD        |  |0      | 00270000 |
135   0   |MUL        | C|958    | 001483BE |   COMM SUBEXP ELIM
136   0   |STO        |  |0      | 00280000 |
137   0   |DEL        |  |0      | 002A0000 |
138   0   |XIT        |  |26     | 0004001A |
139   0   |BRS        |  |0      | 002E0000 |
140   0   |ENT        |  |185    | 000300B9 |
****************************************************************************************
```

```
      FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

 FINAL POOL FOR  BLOCK# 1 IS:
(NULL)


 FINAL POOL FOR  BLOCK# 2 IS:
 A(6,*,210)  A(5,143,143)  A(4,*,211)  A(3,143,143)  A(2,*,212)  A(1,141,141)


 FINAL POOL FOR  BLOCK# 3 IS:
 A(6,*,214)  A(5,143,143)  A(4,*,215)  A(3,143,143)  A(2,*,216)  A(1,141,141)


 FINAL POOL FOR  BLOCK# 4 IS:
 A(6,*,217)  A(5,*,218)  A(4,*,219)  A(3,143,143)  A(2,*,220)  A(1,141,141)


 FINAL POOL FOR  BLOCK# 5 IS:
 A(6,*,177)  A(5,*,178)  A(4,*,179)  A(3,143,143)  A(2,*,180)  A(1,141,141)


 FINAL POOL FOR  BLOCK# 6 IS:
 A(6,*,197)  A(5,*,198)  A(4,*,199)  A(3,143,143)  A(2,*,200)  A(1,141,141)


 FINAL POOL FOR  BLOCK# 7 IS:
 A(6,*,190)  A(5,*,191)  A(4,*,192)  A(3,143,143)  A(2,*,193)  A(1,141,141)


 FINAL POOL FOR  BLOCK# 8 IS:
 A(6,*,186)  A(5,*,187)  A(4,*,188)  A(3,143,143)  A(2,*,189)  A(1,141,141)


 FINAL POOL FOR  BLOCK# 9 IS:
 C(143:174,*,176)  A(6,*,172)  A(5,*,173)  A(4,*,174)  A(3,143,143)
 A(2,*,175)  A(1,141,141)


 FINAL POOL FOR  BLOCK# 10 IS:
 C(141:212,*,213)  A(6,*,210)  A(5,143,143)  A(4,*,211)  A(3,143,143)
 A(2,*,212)  A(1,141,141)
```

```
BASIC BLOCK #1
        BEGINNING AT 0, ENDING AT 26
        BLOCK TRAVERSED 1 TIMES
        0 REFERENCES:
BASIC BLOCK #2
        BEGINNING AT 26, ENDING AT 34
        BLOCK TRAVERSED 2 TIMES
        1 REFERENCES:
              1
BASIC BLOCK #3
        BEGINNING AT 35, ENDING AT 36
        BLOCK TRAVERSED 2 TIMES
        1 REFERENCES:
              2
BASIC BLOCK #4
        BEGINNING AT 36, ENDING AT 44
        BLOCK TRAVERSED 2 TIMES
        1 REFERENCES:
              3
BASIC BLOCK #5
        BEGINNING AT 45, ENDING AT 46
        BLOCK TRAVERSED 2 TIMES
        1 REFERENCES:
              4
BASIC BLOCK #6
        BEGINNING AT 46, ENDING AT 54
        BLOCK TRAVERSED 3 TIMES
        1 REFERENCES:
              5
BASIC BLOCK #7
        BEGINNING AT 55, ENDING AT 78
        BLOCK TRAVERSED 3 TIMES
        1 REFERENCES:
              6
BASIC BLOCK #8
        BEGINNING AT 79, ENDING AT 108
        BLOCK TRAVERSED 2 TIMES
        1 REFERENCES:
              6
BASIC BLOCK #9
        BEGINNING AT 109, ENDING AT 139
        BLOCK TRAVERSED 1 TIMES
        1 REFERENCES:
              4
BASIC BLOCK #10
        BEGINNING AT 140, ENDING AT 140
        BLOCK TRAVERSED 1 TIMES
        1 REFERENCES:
              2

BLOCK SUMMARY DATA:

BLOCK NUMBER                          NUMBER OF PASSES

     1                                       1
     2                                       2
     3                                       2
     4                                       2
     5                                       2
     6                                       3
     7                                       3
     8                                       2
     9                                       1
     10                                      1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 19
USING THE STEEPEST DESCENT (MINIMUM CURRENT POOL)  BLOCK SELECTION ALGORITHM.

TIME FOR THE OPTIMIZATION WAS 0.212 SECONDS.

                    ***CODE SYNTHESIS FILTER IS COMPLETE***
```

```
CARD |BL| SYL |
  1  | 0|   0 |COMMENT     TEST PROGRAM # 2        ;
$EXECUTE
  3  | 0|   1 |COMMENT  SAMPLE OF THE STUDENT PROGRAM NUMBER ONE (#1)
  4  | 0|   1 |         THIS PROGRAM READS INPUT CARDS AND THEN TRANSLATES THEM TO
  5  | 0|   1 |         BASIC MATHEMATICAL OPERATIONS.  THE FIRST NUMBER ON AN INPUT
  6  | 0|   1 |         CARD INDICATES THE OPERATION DESIRED AND THE NEXT TWO NUMBERS
  7  | 0|   1 |         ARE THE OPERAND (X AND Y).  WHEN A CARD IS READ IN WHICH THE
  8  | 0|   1 |         THE MATHEMATICAL OPERATOR CODE IS A 10 THE PROGRAM TERMINATES. ;
  9  | 0|   1 |BEGIN LOCAL X,Y,Z,A,TEMP,HUNDS,INT,TENS;
 10  | 1|   1 |READ (X,Y,Z);
 11  | 1|  10 |WHILE X NEQ 10 DO
 12  | 1|  19 |BEGIN
 13  | 1|  19 |IF X EQL 2 THEN A := 100 * (Y + Z)
 14  | 2|  34 |ELSE
 15  | 2|  37 |IF X EQL 3 THEN A := 100 * Y * Z
 16  | 2|  53 |ELSE
 17  | 2|  58 |IF X EQL 7 THEN A := (100 * Y) / Z
 18  | 2|  74 |ELSE
 19  | 2|  79 |IF X EQL 4 THEN A := 100 * (Y - Z);
 20  | 2| 104 |TEMP := A / 10;
 21  | 2| 111 |HUNDS := A - TEMP * 10;
 22  | 2| 121 |INT := A / 100;
 23  | 2| 128 |TENS := A / 10 - INT * 10;
 24  | 2| 140 |READ (X, Y, Z);
 25  | 2| 149 |END;
 26  | 1| 152 |END
 27  | 1| 152 |EOF
CODE FILE COPIED (152 WORDS)
CONSTANT TABLE COPIED (12 WORDS)
2 RECORDS WRITTEN INTO FILE 1
END OF COMPILATION FEBRUARY 12, 1975.  CLOCK TIME = 20:53:7.88.

27 CARDS WERE READ.
NO ERRORS WERE DETECTED.
```

```
***CODE SYNTHESIS FILTER***

BLOCK SELECTION METHOD WILL BE LAST-IN-FIRST-OUT

***C O M P L E T E   T A B L E   D U M P***
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

***T A B L E   D U M P*** AT LOCATION = 0

CONSTANTS <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
CONSYM   |10  |2   |100 |3   |7   |4
CONTYPE  |INT |INT |INT |INT |INT |INT
CONVAL   |153 |154 |155 |156 |157 |158
CONINT   |10  |2   |100 |3   |7   |4

ADDR TABL <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

VALUE STACK (TOP AT 127)

CONT STK  <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
BLOCK#   |0
ENTRY    |0

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

 ****  FORMATTED INTERMEDIATE CODE DUMP  ****
 LOC OFF    OP CODE    CNS ADR    RAW CODE      OPTIMIZATION
     SET                  /ETC                       TYPE
*************************************************************************
 0    0   |ENT       |C|153     |00038099 |
 1    0   |TOGGLE    | |257     |00080101 |
 2    0   |LOC       | |1       |00020001 |
 3    0   |INT       | |0       |00050000 |
 4    0   |STD       | |0       |00290000 |
 5    0   |LOC       | |2       |00020002 |
 6    0   |INT       | |0       |00050000 |
 7    0   |STD       | |0       |00290000 |
 8    0   |LOC       | |3       |00020003 |
 9    0   |INT       | |0       |00050000 |
 10   0   |STD       | |0       |00290000 |
 11   0   |ENT       | |1       |00030001 |
 12   0   |LOC       | |1       |00020001 |
 13   0   |LOD       | |0       |00270000 |
 14   0   |INT       | |1       |00050001 |
 15   0   |NEQ       | |0       |001F0000 |
 16   0   |XIT       | |152     |00040098 |
 17   0   |XIT       | |19      |00040013 |
 18   0   |BSC       | |0       |002F0000 |
 19   0   |ENT       | |1       |00030001 |
 20   0   |LOC       | |1       |00020001 |
 21   0   |LOD       | |0       |00270000 |
 22   0   |INT       | |2       |00050002 |
 23   0   |EQL       | |0       |001E0000 |
 24   0   |XIT       | |40      |00040028 |
 25   0   |XIT       | |27      |0004001B |
 26   0   |BSC       | |0       |002F0000 |
 27   0   |ENT       | |1       |00030001 |
 28   0   |LOC       | |4       |00020004 |
 29   0   |INT       | |3       |00050003 |
 30   0   |LOC       | |2       |00020002 |
 31   0   |LOD       | |0       |00270000 |
 32   0   |LOC       | |3       |00020003 |
 33   0   |LOD       | |0       |00270000 |
 34   0   |ADD       | |0       |00100000 |
 35   0   |MUL       | |0       |00140000 |
 36   0   |STO       | |0       |00280000 |
 37   0   |DEL       | |0       |002A0000 |
 38   0   |XIT       | |104     |00040068 |
 39   0   |BRS       | |0       |002E0000 |
 40   0   |ENT       | |1       |00030001 |
 41   0   |LOC       | |1       |00020001 |
```

42	0	LOD	0	00270000
43	0	INT	4	00050004
44	0	EQL	0	001E0000
45	0	XIT	61	0004003D
46	0	XIT	48	00040030
47	0	BSC	0	002F0000
48	0	ENT	1	00030001
49	0	LOC	4	00020004
50	0	INT	3	00050003
51	0	LOC	2	00020002
52	0	LOD	0	00270000
53	0	MUL	0	00140000
54	0	LOC	3	00020003
55	0	LOD	0	00270000
56	0	MUL	0	00140000
57	0	STO	0	00280000
58	0	DEL	0	002A0000
59	0	XIT	103	00040067
60	0	BRS	0	002E0000
61	0	ENT	1	00030001
62	0	LOC	1	00020001
63	0	LOD	0	00270000
64	0	INT	5	00050005
65	0	EQL	0	001E0000
66	0	XIT	82	00040052
67	0	XIT	69	00040045
68	0	BSC	0	002F0000
69	0	ENT	1	00030001
70	0	LOC	4	00020004
71	0	INT	3	00050003
72	0	LOC	2	00020002
73	0	LOD	0	00270000
74	0	MUL	0	00140000
75	0	LOC	3	00020003
76	0	LOD	0	00270000
77	0	DIV	0	00160000
78	0	STO	0	00280000
79	0	DEL	0	002A0000
80	0	XIT	102	00040066
81	0	BRS	0	002E0000
82	0	ENT	1	00030001
83	0	LOC	1	00020001
84	0	LOD	0	00270000
85	0	INT	6	00050006
86	0	EQL	0	001E0000
87	0	XIT	101	00040065
88	0	XIT	90	0004005A
89	0	BSC	0	002F0000
90	0	ENT	1	00030001
91	0	LOC	4	00020004
92	0	INT	3	00050003
93	0	LOC	2	00020002
94	0	LOD	0	00270000
95	0	LOC	3	00020003
96	0	LOD	0	00270000
97	0	SUB	0	00120000
98	0	MUL	0	00140000
99	0	STO	0	00280000
100	0	DEL	0	002A0000
101	0	ENT	1	00030001
102	0	ENT	1	00030001
103	0	ENT	1	00030001
104	0	ENT	1	00030001
105	0	LOC	5	00020005
106	0	LOC	4	00020004
107	0	LOD	0	00270000
108	0	INT	1	00050001
109	0	DIV	0	00160000
110	0	STO	0	00280000
111	0	DEL	0	002A0000
112	0	LOC	6	00020006
113	0	LOC	4	00020004
114	0	LOD	0	00270000
115	0	LOC	5	00020005
116	0	LOD	0	00270000
117	0	INT	1	00050001
118	0	MUL	0	00140000
119	0	SUB	0	00120000
120	0	STO	0	00280000
121	0	DEL	0	002A0000
122	0	LOC	7	00020007
123	0	LOC	4	00020004
124	0	LOD	0	00270000
125	0	INT	3	00050003
126	0	DIV	0	00160000
127	0	STO	0	00280000
128	0	DEL	0	002A0000
129	0	LOC	8	00020008

```
130  0  |LOC       | |4      | 00020004 |
131  0  |LOD       | |0      | 00270000 |
132  0  |INT       | |1      | 00050001 |
133  0  |DIV       | |0      | 00160000 |
134  0  |LOC       | |7      | 00020007 |
135  0  |LOD       | |0      | 00270000 |
136  0  |INT       | |1      | 00050001 |
137  0  |MUL       | |0      | 00140000 |
138  0  |SUB       | |0      | 00120000 |
139  0  |STO       | |0      | 00280000 |
140  0  |DEL       | |0      | 002A0000 |
141  0  |LOC       | |1      | 00020001 |
142  0  |INT       | |0      | 00050000 |
143  0  |STD       | |0      | 00290000 |
144  0  |LOC       | |2      | 00020002 |
145  0  |INT       | |0      | 00050000 |
146  0  |STD       | |0      | 00290000 |
147  0  |LOC       | |3      | 00020003 |
148  0  |INT       | |0      | 00050000 |
149  0  |STD       | |0      | 00290000 |
150  0  |XIT       | |11     | 00040008 |
151  0  |BRS       | |0      | 002E0000 |
152  0  |ENT       | |1      | 00030001 |
****************************************************************************************

      BASIC BLOCK #1
            BEGINNING AT 0, ENDING AT 0
            BLOCK TRAVERSED 0 TIMES
            0 REFERENCES:

***E N D   O F   C O M P L E T E   T A B L E   D U M P***

B+2   0  |LOC       | |1      | 00020001 |
B+3   0  |INT       | |0      | 00050000 |
B+4   0  |STD       | |0      | 00290000 |
B+5   0  |LOC       | |2      | 00020002 |
B+6   0  |INT       | |0      | 00050000 |
B+7   0  |STD       | |0      | 00290000 |
B+8   0  |LOC       | |3      | 00020003 |
B+9   0  |INT       | |0      | 00050000 |
B+10  0  |STD       | |0      | 00290000 |
B+11  0  |ENT       | |1      | 00030001 |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 2 IS:
 A(3,*,161)  A(2,*,160)  A(1,*,159)

 CONTROL AT 11, FROM BLK 1 TO BLK 2 (PASS 1)
C+11  0  |ENT       |C|157    | 0003809D |
B+12  0  |LOC       | |1      | 00020001 |
B+13  0  |LOD       | |0      | 00270000 |
B+14  0  |INT       | |1      | 00050001 |
B+15  0  |NEQ       | |0      | 001F0000 |
B+16  0  |XIT       | |152    | 00040098 |
B+17  0  |XIT       | |19     | 00040013 |
B+18  0  |BSC       | |0      | 002F0000 |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 3 IS:
 C(159:153,*,162)  A(3,*,161)  A(2,*,160)  A(1,*,159)

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 4 IS:
 C(159:153,*,162)  A(3,*,161)  A(2,*,160)  A(1,*,159)

 CONTROL AT 19, FROM BLK 2 TO BLK 4 (PASS 1)
C+19  0  |ENT       |C|167    | 000380A7 |
B+20  0  |LOC       | |1      | 00020001 |
B+21  0  |LOD       | |0      | 00270000 |
B+22  0  |INT       | |2      | 00050002 |
B+23  0  |EQL       | |0      | 001E0000 |
B+24  0  |XIT       | |40     | 00040028 |
B+25  0  |XIT       | |27     | 0004001B |
B+26  0  |BSC       | |0      | 002F0000 |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 5 IS:
 C(159:153,*,162)  C(159:154,*,163)  A(3,*,161)  A(2,*,160)  A(1,*,159)

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 6 IS:
 C(159:153,*,162)  C(159:154,*,163)  A(3,*,161)  A(2,*,160)  A(1,*,159)

 CONTROL AT 27, FROM BLK 4 TO BLK 6 (PASS 1)
C+27  0  |ENT       |C|177    | 000380B1 |
B+28  0  |LOC       | |4      | 00020004 |
B+29  0  |INT       | |3      | 00050003 |
B+30  0  |LOC       | |2      | 00020002 |
B+31  0  |LOD       | |0      | 00270000 |
B+32  0  |LOC       | |3      | 00020003 |
```

```
B+33   0   |LOD            | |0        | 00270000 |
B+34   0   |ADD            | |0        | 00100000 |
B+35   0   |MUL            | |0        | 00140000 |
B+36   0   |STO            | |0        | 00280000 |
B+37   0   |DEL            | |0        | 002A0000 |
B+38   0   |XIT            | |104      | 00040068 |
B+39   0   |BRS            | |0        | 002E0000 |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 7 IS:
 C(159:154,*,163)  C(159:153,*,162)  C(160:161,*,164)  C(155:164,*,165)
 A(4,*,165)  A(3,*,161)  A(2,*,160)  A(1,*,159)

 CONTROL AT 104, FROM BLK 6 TO BLK 7 (PASS 1)
C+104  0   |ENT            |C|182      | 00038086 |
B+105  0   |LOC            | |5        | 00020005 |
B+106  0   |LOC            | |4        | 00020004 |
B+107  0   |LOD            | |0        | 00270000 |
B+108  0   |INT            | |1        | 00050001 |
B+109  0   |DIV            | |0        | 00160000 |
B+110  0   |STO            | |0        | 00280000 |
B+111  0   |DEL            | |0        | 002A0000 |
B+112  0   |LOC            | |6        | 00020006 |
B+113  0   |LOC            | |4        | 00020004 |
B+114  0   |LOD            | |0        | 00270000 |
B+115  0   |LOC            | |5        | 00020005 |
B+116  0   |LOD            | |0        | 00270000 |
B+117  0   |INT            | |1        | 00050001 |
B+118  0   |MUL            | |0        | 00140000 |
B+119  0   |SUB            | |0        | 00120000 |
B+120  0   |STO            | |0        | 00280000 |
B+121  0   |DEL            | |0        | 002A0000 |
B+122  0   |LOC            | |7        | 00020007 |
B+123  0   |LOC            | |4        | 00020004 |
B+124  0   |LOD            | |0        | 00270000 |
B+125  0   |INT            | |3        | 00050003 |
B+126  0   |DIV            | |0        | 00160000 |
B+127  0   |STO            | |0        | 00280000 |
B+128  0   |DEL            | |0        | 002A0000 |
B+129  0   |LOC            | |8        | 00020008 |
B+130  0   |LOC            | |4        | 00020004 |
B+131  0   |LOD            | |0        | 00270000 |
B+132  0   |INT            | |1        | 00050001 |
B+133  0   |DIV            | |0        | 00160000 |
B+134  0   |LOC            | |7        | 00020007 |
B+135  0   |LOD            | |0        | 00270000 |
B+136  0   |INT            | |1        | 00050001 |
B+137  0   |MUL            | |0        | 00140000 |
B+138  0   |SUB            | |0        | 00120000 |
B+139  0   |STO            | |0        | 00280000 |
B+140  0   |DEL            | |0        | 002A0000 |
B+141  0   |LOC            | |1        | 00020001 |
B+142  0   |INT            | |0        | 00050000 |
B+143  0   |STD            | |0        | 00290000 |
B+144  0   |LOC            | |2        | 00020002 |
B+145  0   |INT            | |0        | 00050000 |
B+146  0   |STD            | |0        | 00290000 |
B+147  0   |LOC            | |3        | 00020003 |
B+148  0   |INT            | |0        | 00050000 |
B+149  0   |STD            | |0        | 00290000 |
B+150  0   |XIT            | |11       | 0004000B |
B+151  0   |BRS            | |0        | 002E0000 |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 7 IS:
 C(155:164,*,165)  C(160:161,*,164)  C(159:153,*,162)  C(159:154,*,163)
 C(159:153,*,163)  C(165:153,*,166)  C(166:153,*,167)  C(155:164,*,167)
 C(165:167,*,168)  C(165:155,*,169)  C(169:153,*,170)  C(165:155,*,170)
 C(166:170,*,171)  A(3,*,174)  A(2,*,173)  A(1,*,172)  A(8,*,171)  A(7,*,169)
 A(6,*,168)  A(5,*,166)  A(4,*,165)


 CURRENT POOL FOR BLOCK# 2 IS:
 A(3,*,161)  A(2,*,160)  A(1,*,159)

AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

 CONTROL AT 40, FROM BLK 4 TO BLK 5 (PASS 1)
C+40   0   |ENT            |C|172      | 000380AC |
B+41   0   |LOC            | |1        | 00020001 |
B+42   0   |LOD            | |0        | 00270000 |
B+43   0   |INT            | |4        | 00050004 |
B+44   0   |EQL            | |0        | 001E0000 |
B+45   0   |XIT            | |61       | 0004003D |
B+46   0   |XIT            | |48       | 00040030 |
B+47   0   |BSC            | |0        | 002F0000 |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 5 IS:
```

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 8 IS:
C(159:154,*,163) C(159:153,*,162) C(159:156,*,178) A(3,*,161) A(2,*,160)
A(1,*,159)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 9 IS:
C(159:154,*,163) C(159:153,*,162) C(159:156,*,178) A(3,*,161) A(2,*,160)
A(1,*,159)

CONTROL AT 48, FROM BLK 5 TO BLK 9 (PASS 1)

C+48	0	ENT	C	192	000380C0	
B+49	0	LOC		4	00020004	
B+50	0	INT		3	00050003	
B+51	0	LOC		2	00020002	
B+52	0	LOD		0	00270000	
B+53	0	MUL		0	00140000	
B+54	0	LOC		3	00020003	
B+55	0	LOD		0	00270000	
B+56	0	MUL		0	00140000	
B+57	0	STO		0	00280000	
B+58	0	DEL		0	002A0000	
B+59	0	XIT		103	00040067	
B+60	0	BRS		0	002E0000	

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 10 IS:
C(159:156,*,178) C(159:153,*,162) C(159:154,*,163) C(155:160,*,179)
C(179:161,*,180) A(4,*,180) A(3,*,161) A(2,*,160) A(1,*,159)

CONTROL AT 103, FROM BLK 9 TO BLK 10 (PASS 1)

C+103	0	ENT	C	197	000380C5	
B+104	0	ENT	C	182	000380B6	

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 10 IS:
C(179:161,*,180) C(155:160,*,179) C(159:154,*,163) C(159:153,*,162)
C(159:154,*,162) C(159:156,*,178) A(4,*,180) A(3,*,161) A(2,*,160)
A(1,*,159)

CURRENT POOL FOR BLOCK# 7 IS:
C(159:154,*,163) C(159:153,*,162) C(160:161,*,164) C(155:164,*,165)
A(4,*,165) A(3,*,161) A(2,*,160) A(1,*,159)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 7 IS:
A(4,*,181) A(3,*,161) A(2,*,160) A(1,*,159)

CONTROL AT 104, FROM BLK 10 TO BLK 7 (PASS 2)

C+104	0	ENT	C	182	000380B6	
B+105	0	LOC		5	00020005	
B+106	0	LOC		4	00020004	
B+107	0	LOD		0	00270000	
B+108	0	INT		1	00050001	
B+109	0	DIV	C	988	001683DC	
B+110	0	STO		0	00280000	
B+111	0	DEL		0	002A0000	
B+112	0	LOC		6	00020006	
B+113	0	LOC		4	00020004	
B+114	0	LOD		0	00270000	
B+115	0	LOC		5	00020005	
B+116	0	LOD		0	00270000	
B+117	0	INT		1	00050001	
B+118	0	MUL	C	984	001483D8	
B+119	0	SUB	C	980	001283D4	
B+120	0	STO		0	00280000	
B+121	0	DEL		0	002A0000	
B+122	0	LOC		7	00020007	
B+123	0	LOC		4	00020004	
B+124	0	LOD		0	00270000	
B+125	0	INT		3	00050003	
B+126	0	DIV	C	976	001683D0	
B+127	0	STO		0	00280000	
B+128	0	DEL		0	002A0000	
B+129	0	LOC		8	00020008	
B+130	0	LOC		4	00020004	
B+131	0	LOD		0	00270000	
B+132	0	INT		1	00050001	
B+133	0	DIV	C	972	001683CC	COMM SUBEXP ELIM
B+134	0	LOC		7	00020007	
B+135	0	LOD		0	00270000	
B+136	0	INT		1	00050001	
B+137	0	MUL	C	968	001483C8	
B+138	0	SUB	C	964	001283C4	
B+139	0	STO		0	00280000	
B+140	0	DEL		0	002A0000	

```
B+141  0  |LOC   | |1    | 00020001 |
B+142  0  |INT   | |0    | 00050000 |
B+143  0  |STD   | |0    | 00290000 |
B+144  0  |LOC   | |2    | 00020002 |
B+145  0  |INT   | |0    | 00050000 |
B+146  0  |STD   | |0    | 00290000 |
B+147  0  |LOC   | |3    | 00020003 |
B+148  0  |INT   | |0    | 00050000 |
B+149  0  |STD   | |0    | 00290000 |
B+150  0  |XIT   | |11   | 0004000B |
B+151  0  |BRS   | |0    | 002E0000 |
BEFORE THE MEET OPERATION:

 OUTPUT PCOL FROM BLOCK# 7 IS:
 C(181:153,*,182)  C(182:153,*,183)  C(181:183,*,184)  C(181:155,*,185)
 C(185:153,*,186)  C(181:155,*,186)  C(182:186,*,187)  A(3,*,190)  A(2,*,189)
 A(1,*,188)  A(8,*,187)  A(7,*,185)  A(6,*,184)  A(5,*,182)  A(4,*,181)


 CURRENT POOL FOR BLOCK# 2 IS:
 A(3,*,175)  A(2,*,176)  A(1,*,177)

AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

 CONTROL AT 61, FROM BLK 5 TO BLK 8 (PASS 1)
C+61   0  |ENT   |C|187  | 000380BB |
B+62   0  |LOC   | |1    | 00020001 |
B+63   0  |LOD   | |0    | 00270000 |
B+64   0  |INT   | |5    | 00050005 |
B+65   0  |EQL   | |0    | 001E0000 |
B+66   0  |XIT   | |82   | 00040052 |
B+67   0  |XIT   | |69   | 00040045 |
B+68   0  |BSC   | |0    | 002F0000 |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 11 IS:
 C(159:156,*,178)  C(159:153,*,162)  C(159:154,*,163)  C(159:157,*,194)
 A(3,*,161)  A(2,*,160)  A(1,*,159)


 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 12 IS:
 C(159:156,*,178)  C(159:155,*,162)  C(159:154,*,163)  C(159:157,*,194)
 A(3,*,161)  A(2,*,160)  A(1,*,159)

 CONTROL AT 69, FROM BLK 8 TO BLK 12 (PASS 1)
C+69   0  |ENT   |C|207  | 000380CF |
B+70   0  |LOC   | |4    | 00020004 |
B+71   0  |INT   | |3    | 00050003 |
B+72   0  |LOC   | |2    | 00020002 |
B+73   0  |LOD   | |0    | 00270000 |
B+74   0  |MUL   | |0    | 00140000 |
B+75   0  |LOC   | |3    | 00020003 |
B+76   0  |LOD   | |0    | 00270000 |
B+77   0  |DIV   | |0    | 00160000 |
B+78   0  |STO   | |0    | 00280000 |
B+79   0  |DEL   | |0    | 002A0000 |
B+80   0  |XIT   | |102  | 00040066 |
B+81   0  |BRS   | |0    | 002E0000 |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 13 IS:
 C(159:157,*,194)  C(159:154,*,163)  C(159:153,*,162)  C(159:156,*,178)
 C(155:160,*,195)  C(195:161,*,196)  A(4,*,196)  A(3,*,161)  A(2,*,160)
 A(1,*,159)

 CONTROL AT 102, FROM BLK 12 TO BLK 13 (PASS 1)
C+102  0  |ENT   |C|212  | 000380D4 |
B+103  0  |ENT   |C|197  | 000380C5 |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 13 IS:
 C(195:161,*,196)  C(155:160,*,195)  C(159:156,*,178)  C(159:153,*,162)
 C(159:154,*,163)  C(159:153,*,163)  C(159:157,*,194)  A(4,*,196)  A(3,*,161)
 A(2,*,160)  A(1,*,159)


 CURRENT POOL FOR BLOCK# 10 IS:
 C(159:156,*,178)  C(159:153,*,162)  C(159:154,*,163)  C(155:160,*,179)
 C(179:161,*,180)  A(4,*,180)  A(3,*,161)  A(2,*,160)  A(1,*,159)

AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 10 IS:
 A(4,*,197)  A(3,*,161)  A(2,*,160)  A(1,*,159)


 CONTROL AT 103, FROM BLK 13 TO BLK 10 (PASS 2)
```

```
C+103  0    |ENT          |C|197    | 000380C5  |
B+104  0    |ENT          |C|182    | 00038086  |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 10 IS:
 A(4,*,197)  A(3,*,161)  A(2,*,160)  A(1,*,159)


 CURRENT POOL FOR BLOCK# 7 IS:
 A(4,*,181)  A(3,*,161)  A(2,*,160)  A(1,*,159)

AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

 CONTROL AT 82, FROM BLK 8 TO BLK 11 (PASS 1)
C+82   0    |ENT          |C|202    | 000380CA  |
B+83   0    |LOC          | |1      | 00020001  |
B+84   0    |LOD          | |0      | 00270000  |
B+85   0    |INT          | |6      | 00050006  |
B+86   0    |EQL          | |0      | 001E0000  |
B+87   0    |XIT          | |101    | 00040065  |
B+88   0    |XIT          | |90     | 0004005A  |
B+89   0    |BSC          | |0      | 002F0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 14 IS:
 C(159:157,*,194)  C(159:154,*,163)  C(159:153,*,162)  C(159:156,*,178)
 C(159:158,*,199)  A(3,*,161)  A(2,*,160)  A(1,*,159)


 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 15 IS:
 C(159:157,*,194)  C(159:154,*,163)  C(159:153,*,162)  C(159:156,*,178)
 C(159:158,*,199)  A(3,*,161)  A(2,*,160)  A(1,*,159)

 CONTROL AT 90, FROM BLK 11 TO BLK 15 (PASS 1)
C+90   0    |ENT          |C|222    | 000380DE  |
B+91   0    |LOC          | |4      | 00020004  |
B+92   0    |INT          | |3      | 00050003  |
B+93   0    |LOC          | |2      | 00020002  |
B+94   0    |LOD          | |0      | 00270000  |
B+95   0    |LOC          | |3      | 00020003  |
B+96   0    |LOD          | |0      | 00270000  |
B+97   0    |SUB          | |0      | 00120000  |
B+98   0    |MUL          | |0      | 00140000  |
B+99   0    |STO          | |0      | 00280000  |
B+100  0    |DEL          | |0      | 002A0000  |
B+101  0    |ENT          |C|217    | 000380D9  |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 15 IS:
 C(159:158,*,199)  C(159:156,*,178)  C(159:153,*,162)  C(159:154,*,163)
 C(159:153,*,163)  C(159:157,*,194)  C(160:161,*,200)  C(155:200,*,201)
 A(4,*,201)  A(3,*,161)  A(2,*,160)  A(1,*,159)


 CURRENT POOL FOR BLOCK# 14 IS:
 C(159:157,*,194)  C(159:154,*,163)  C(159:153,*,162)  C(159:156,*,178)
 C(159:158,*,199)  A(3,*,161)  A(2,*,160)  A(1,*,159)
AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 14 IS:
 A(3,*,161)  A(2,*,160)  A(1,*,159)


 CONTROL AT 101, FROM BLK 15 TO BLK 14 (PASS 1)
C+101  0    |ENT          |C|217    | 000380D9  |
B+102  0    |ENT          |C|212    | 000380D4  |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 14 IS:
 A(3,*,161)  A(2,*,160)  A(1,*,159)


 CURRENT POOL FOR BLOCK# 13 IS:
 C(159:157,*,194)  C(159:154,*,163)  C(159:153,*,162)  C(159:156,*,178)
 C(155:160,*,195)  C(195:161,*,196)  A(4,*,196)  A(3,*,161)  A(2,*,160)
 A(1,*,159)
AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 13 IS:
 A(3,*,161)  A(2,*,160)  A(1,*,159)


 CONTROL AT 102, FROM BLK 14 TO BLK 13 (PASS 2)
```

```
C+102   0    |ENT            |C|212     | 000380D4   |
B+103   0    |ENT            |C|197     | 000380C5   |

BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 13 IS:
 A(3,*,161)  A(2,*,160)  A(1,*,159)


 CURRENT POOL FOR BLOCK# 10 IS:
 A(4,*,197)  A(3,*,161)  A(2,*,160)  A(1,*,159)

AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 10 IS:
 A(3,*,161)  A(2,*,160)  A(1,*,159)


 CONTROL AT 103, FROM BLK 13 TO BLK 10 (PASS 3)
C+103   0    |ENT            |C|197     | 000380C5   |
B+104   0    |ENT            |C|182     | 00038086   |

BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 10 IS:
 A(3,*,161)  A(2,*,160)  A(1,*,159)


 CURRENT POOL FOR BLOCK# 7 IS:
 A(4,*,198)  A(3,*,161)  A(2,*,160)  A(1,*,159)

AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 7 IS:
 A(3,*,161)  A(2,*,160)  A(1,*,159)


 CONTROL AT 104, FROM BLK 10 TO BLK 7 (PASS 3)
C+104   0    |ENT            |C|182     | 00038086   |
B+105   0    |LOC            | |5       | 00020005   |
B+106   0    |LOC            | |4       | 00020004   |
B+107   0    |LOD            | |0       | 00270000   |
B+108   0    |INT            | |1       | 00050001   |
B+109   0    |DIV            |C|988     | 001683DC   |
B+110   0    |STO            | |0       | 00280000   |
B+111   0    |DEL            | |0       | 002A0000   |
B+112   0    |LOC            | |6       | 00020006   |
B+113   0    |LOC            | |4       | 00020004   |
B+114   0    |LOD            | |0       | 00270000   |
B+115   0    |LOC            | |5       | 00020005   |
B+116   0    |LOD            | |0       | 00270000   |
B+117   0    |INT            | |1       | 00050001   |
B+118   0    |MUL            |C|984     | 001483D8   |
B+119   0    |SUB            |C|980     | 001283D4   |
B+120   0    |STO            | |0       | 00280000   |
B+121   0    |DEL            | |0       | 002A0000   |
B+122   0    |LOC            | |7       | 00020007   |
B+123   0    |LOC            | |4       | 00020004   |
B+124   0    |LOD            | |0       | 00270000   |
B+125   0    |INT            | |3       | 00050003   |
B+126   0    |DIV            |C|976     | 001683D0   |
B+127   0    |STO            | |0       | 00280000   |
B+128   0    |DEL            | |0       | 002A0000   |
B+129   0    |LOC            | |8       | 00020008   |
B+130   0    |LOC            | |4       | 00020004   |
B+131   0    |LOD            | |0       | 00270000   |
B+132   0    |INT            | |1       | 00050001   |
B+133   0    |DIV            |C|972     | 001683CC   |   COMM SUBEXP ELIM
B+134   0    |LOC            | |7       | 00020007   |
B+135   0    |LOD            | |0       | 00270000   |
B+136   0    |INT            | |1       | 00050001   |
B+137   0    |MUL            |C|968     | 001483C8   |
B+138   0    |SUB            |C|964     | 001283C4   |
B+139   0    |STO            | |0       | 00280000   |
B+140   0    |DEL            | |0       | 002A0000   |
B+141   0    |LOC            | |1       | 00020001   |
B+142   0    |INT            | |0       | 00050000   |
B+143   0    |STD            | |0       | 00290000   |
B+144   0    |LOC            | |2       | 00020002   |
B+145   0    |INT            | |0       | 00050000   |
B+146   0    |STD            | |0       | 00290000   |
B+147   0    |LOC            | |3       | 00020003   |
B+148   0    |INT            | |0       | 00050000   |
B+149   0    |STD            | |0       | 00290000   |
B+150   0    |XIT            | |11      | 00040008   |
B+151   0    |BRS            | |0       | 002E0000   |

BEFORE THE MEET OPERATION:
```

```
 OUTPUT POOL FROM BLOCK# 7 IS:
 C(202:153,*,203)  C(203:153,*,204)  C(202:204,*,205)  C(202:155,*,206)
 C(206:153,*,207)  C(202:155,*,207)  C(203:207,*,208)  A(3,*,211)  A(2,*,210)
 A(1,*,209)  A(8,*,208)  A(7,*,206)  A(6,*,205)  A(4,*,202)  A(5,*,203)


 CURRENT POOL FOR BLOCK# 2 IS:
 A(3,*,191)  A(2,*,192)  A(1,*,193)

AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

 CONTROL AT 152, FROM BLK 2 TO BLK 3 (PASS 1)
C+152  0   |ENT          |C|162   | 000380A2  |

                                    FINAL OPTIMIZATION RESULTS
##########################################################################################
------------------------------------------------------------------------------------------

OPTIMIZATION AT 133
          (153,202|109,203)    COMM SUBEXP ELIM

 ****  FORMATTED INTERMEDIATE CODE DUMP  ****
 LOC OFF     OP CODE     CNS ADR      RAW CODE        OPTIMIZATION
     SET                  /ETC                            TYPE
*************************************************************************
 0     0    |ENT         | |153   | 00030099  |
 1     0    |TOGGLE      | |257   | 00080101  |
 2     0    |LOC         | |1     | 00020001  |
 3     0    |INT         | |0     | 00050000  |
 4     0    |STD         | |0     | 00290000  |
 5     0    |LOC         | |2     | 00020002  |
 6     0    |INT         | |0     | 00050000  |
 7     0    |STD         | |0     | 00290000  |
 8     0    |LOC         | |3     | 00020003  |
 9     0    |INT         | |0     | 00050000  |
 10    0    |STD         | |0     | 00290000  |
 11    0    |ENT         | |157   | 0003009D  |
 12    0    |LOC         | |1     | 00020001  |
 13    0    |LOD         | |0     | 00270000  |
 14    0    |INT         | |1     | 00050001  |
 15    0    |NEQ         | |1024  | 001F0400  |
 16    0    |XIT         | |152   | 00040098  |
 17    0    |XIT         | |19    | 00040013  |
 18    0    |BSC         | |0     | 002F0000  |
 19    0    |ENT         | |167   | 000300A7  |
 20    0    |LOC         | |1     | 00020001  |
 21    0    |LOD         | |0     | 00270000  |
 22    0    |INT         | |2     | 00050002  |
 23    0    |EQL         | |1016  | 001E03F8  |
 24    0    |XIT         | |40    | 00040028  |
 25    0    |XIT         | |27    | 0004001B  |
 26    0    |BSC         | |0     | 002F0000  |
 27    0    |ENT         | |177   | 000300B1  |
 28    0    |LOC         | |4     | 00020004  |
 29    0    |INT         | |3     | 00050003  |
 30    0    |LOC         | |2     | 00020002  |
 31    0    |LOD         | |0     | 00270000  |
 32    0    |LOC         | |3     | 00020003  |
 33    0    |LOD         | |0     | 00270000  |
 34    0    |ADD         | |1004  | 001003EC  |
 35    0    |MUL         | |1000  | 001403E8  |
 36    0    |STO         | |0     | 00280000  |
 37    0    |DEL         | |0     | 002A0000  |
 38    0    |XIT         | |104   | 00040068  |
 39    0    |BRS         | |0     | 002E0000  |
 40    0    |ENT         | |172   | 000300AC  |
 41    0    |LOC         | |1     | 00020001  |
 42    0    |LOD         | |0     | 00270000  |
 43    0    |INT         | |4     | 00050004  |
 44    0    |EQL         | |960   | 001E03C0  |
 45    0    |XIT         | |61    | 0004003D  |
 46    0    |XIT         | |48    | 00040030  |
 47    0    |BSC         | |0     | 002F0000  |
 48    0    |ENT         | |192   | 000300C0  |
 49    0    |LOC         | |4     | 00020004  |
 50    0    |INT         | |3     | 00050003  |
 51    0    |LOC         | |2     | 00020002  |
 52    0    |LOD         | |0     | 00270000  |
 53    0    |MUL         | |944   | 001403B0  |
 54    0    |LOC         | |3     | 00020003  |
 55    0    |LOD         | |0     | 00270000  |
 56    0    |MUL         | |940   | 001403AC  |
 57    0    |STO         | |0     | 00280000  |
```

58	0	DEL		0	002A0000	
59	0	XIT		103	00040067	
60	0	BRS		0	002E0000	
61	0	ENT		187	000300BB	
62	0	LOC		1	00020001	
63	0	LOD		0	00270000	
64	0	INT		5	00050005	
65	0	EQL		926	001E039E	
66	0	XIT		82	00040052	
67	0	XIT		69	00040045	
68	0	BSC		0	002F0000	
69	0	ENT		207	000300CF	
70	0	LOC		4	00020004	
71	0	INT		3	00050003	
72	0	LOC		2	00020002	
73	0	LOD		0	00270000	
74	0	MUL		906	0014038A	
75	0	LOC		3	00020003	
76	0	LOD		0	00270000	
77	0	DIV		902	00160386	
78	0	STO		0	00280000	
79	0	DEL		0	002A0000	
80	0	XIT		102	00040066	
81	0	BRS		0	002E0000	
82	0	ENT		202	000300CA	
83	0	LOC		1	00020001	
84	0	LOD		0	00270000	
85	0	INT		6	00050006	
86	0	EQL		886	001E0376	
87	0	XIT		101	00040065	
88	0	XIT		90	0004005A	
89	0	BSC		0	002F0000	
90	0	ENT		222	000300DE	
91	0	LOC		4	00020004	
92	0	INT		3	00050003	
93	0	LOC		2	00020002	
94	0	LOD		0	00270000	
95	0	LOC		3	00020003	
96	0	LOD		0	00270000	
97	0	SUB		862	0012035E	
98	0	MUL		858	0014035A	
99	0	STO		0	00280000	
100	0	DEL		0	002A0000	
101	0	ENT		217	000300D9	
102	0	ENT		212	000300D4	
103	0	ENT		197	000300C5	
104	0	ENT		182	000300B6	
105	0	LOC		5	00020005	
106	0	LOC		4	00020004	
107	0	LOD		0	00270000	
108	0	INT		1	00050001	
109	0	DIV		988	001603DC	
110	0	STO		0	00280000	
111	0	DEL		0	002A0000	
112	0	LOC		6	00020006	
113	0	LOC		4	00020004	
114	0	LOD		0	00270000	
115	0	LOC		5	00020005	
116	0	LOD		0	00270000	
117	0	INT		1	00050001	
118	0	MUL		984	001403D8	
119	0	SUB		980	001203D4	
120	0	STO		0	00280000	
121	0	DEL		0	002A0000	
122	0	LOC		7	00020007	
123	0	LOC		4	00020004	
124	0	LOD		0	00270000	
125	0	INT		3	00050003	
126	0	DIV		976	001603D0	
127	0	STO		0	00280000	
128	0	DEL		0	002A0000	
129	0	LOC		8	00020008	
130	0	LOC		4	00020004	
131	0	LOD		0	00270000	
132	0	INT		1	00050001	
133	0	DIV	C	972	001683CC	COMM SUBEXP ELIM
134	0	LOC		7	00020007	
135	0	LOD		0	00270000	
136	0	INT		1	00050001	
137	0	MUL		968	001403C8	
138	0	SUB		964	001203C4	
139	0	STO		0	00280000	
140	0	DEL		0	002A0000	
141	0	LOC		1	00020001	
142	0	INT		0	00050000	
143	0	STD		0	00290000	
144	0	LOC		2	00020002	
145	0	INT		0	00050000	

```
146   0   |STD        |  |0     | 00290000 |
147   0   |LOC        |  |3     | 00020003 |
148   0   |INT        |  |0     | 00050000 |
149   0   |STD        |  |0     | 00290000 |
150   0   |XIT        |  |11    | 0004000B |
151   0   |BRS        |  |0     | 002E0000 |
152   0   |ENT        |  |162   | 000300A2 |
***********************************************************************************************

        FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

 FINAL POCL FOR  BLOCK# 1 IS:
(NULL)


 FINAL POCL FOR  BLOCK# 2 IS:
 A(3,*,212)  A(2,*,213)  A(1,*,214)


 FINAL POCL FOR  BLOCK# 3 IS:
 C(159:153,*,162)  A(3,*,161)  A(2,*,160)  A(1,*,159)


 FINAL POCL FOR  BLOCK# 4 IS:
 C(159:153,*,162)  A(3,*,161)  A(2,*,160)  A(1,*,159)


 FINAL POCL FOR  BLOCK# 5 IS:
 C(159:153,*,162)  C(159:154,*,163)  A(3,*,161)  A(2,*,160)  A(1,*,159)


 FINAL POOL FOR  BLOCK# 6 IS:
 C(159:153,*,162)  C(159:154,*,163)  A(3,*,161)  A(2,*,160)  A(1,*,159)


 FINAL POCL FOR  BLOCK# 7 IS:
 A(3,*,161)  A(2,*,160)  A(1,*,159)


 FINAL POCL FOR  BLOCK# 8 IS:
 C(159:154,*,163)  C(159:153,*,162)  C(159:156,*,178)  A(3,*,161)  A(2,*,160)
 A(1,*,159)


 FINAL POCL FOR  BLOCK# 9 IS:
 C(159:154,*,163)  C(159:153,*,162)  C(159:156,*,178)  A(3,*,161)  A(2,*,160)
 A(1,*,159)


 FINAL POCL FOR  BLOCK# 10 IS:
 A(3,*,161)  A(2,*,160)  A(1,*,159)


 FINAL POCL FOR  BLOCK# 11 IS:
 C(159:156,*,178)  C(159:153,*,162)  C(159:154,*,163)  C(159:157,*,194)
 A(3,*,161)  A(2,*,160)  A(1,*,159)


 FINAL POCL FOR  BLOCK# 12 IS:
 C(159:156,*,178)  C(159:153,*,162)  C(159:154,*,163)  C(159:157,*,194)
 A(3,*,161)  A(2,*,160)  A(1,*,159)


 FINAL POCL FOR  BLOCK# 13 IS:
 A(3,*,161)  A(2,*,160)  A(1,*,159)


 FINAL POCL FOR  BLOCK# 14 IS:
 A(3,*,161)  A(2,*,160)  A(1,*,159)
```

```
FINAL POCL FOR  BLOCK# 15 IS:
C(159:157,*,194)  C(159:154,*,163)  C(159:153,*,162)  C(159:156,*,178)
C(159:158,*,199)  A(3,*,161)  A(2,*,160)  A(1,*,159)


     BASIC BLOCK #1
           BEGINNING AT 0, ENDING AT 11
           BLOCK TRAVERSED 1 TIMES
           0 REFERENCES:
     BASIC BLOCK #2
           BEGINNING AT 11, ENDING AT 18
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
              1
     BASIC BLOCK #3
           BEGINNING AT 152, ENDING AT 152
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
              2
     BASIC BLOCK #4
           BEGINNING AT 19, ENDING AT 26
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
              2
     BASIC BLOCK #5
           BEGINNING AT 40, ENDING AT 47
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
              4
     BASIC BLOCK #6
           BEGINNING AT 27, ENDING AT 39
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
              4
     BASIC BLOCK #7
           BEGINNING AT 104, ENDING AT 151
           BLOCK TRAVERSED 3 TIMES
           1 REFERENCES:
              6
     BASIC BLOCK #8
           BEGINNING AT 61, ENDING AT 68
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
              5
     BASIC BLOCK #9
           BEGINNING AT 48, ENDING AT 60
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
              5
     BASIC BLOCK #10
           BEGINNING AT 103, ENDING AT 104
           BLOCK TRAVERSED 3 TIMES
           1 REFERENCES:
              9
     BASIC BLOCK #11
           BEGINNING AT 82, ENDING AT 89
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
              8
     BASIC BLOCK #12
           BEGINNING AT 69, ENDING AT 81
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
              8
     BASIC BLOCK #13
           BEGINNING AT 102, ENDING AT 103
           BLOCK TRAVERSED 2 TIMES
           1 REFERENCES:
             12
     BASIC BLOCK #14
           BEGINNING AT 101, ENDING AT 102
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
             15
     BASIC BLOCK #15
           BEGINNING AT 90, ENDING AT 101
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
             11

   BLOCK SUMMARY DATA:

   BLOCK NUMBER                         NUMBER OF PASSES

       1                                    1
```

2	1
3	1
4	1
5	1
6	1
7	3
8	1
9	1
10	3
11	1
12	1
13	2
14	1
15	1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 20
USING THE LAST-IN-FIRST-OUT BLOCK SELECTION ALGORITHM.

TIME FOR THE OPTIMIZATION WAS 1.411 SECONDS.

CODE SYNTHESIS FILTER IS COMPLETE

```
CARD |BL|  SYL |
   1 | 0|    0 |COMMENT     TEST PROGRAM # 3          ;
$EXECUTE
   3 | 0|    1 |BEGIN LOCAL   A,B,C,D,E,F,G,H,I;
   4 | 1|    1 |B := 5;
   5 | 1|    5 |C := 10;
   6 | 1|    9 |A := B * C;
   7 | 1|   17 |IF A GTR B THEN
   8 | 1|   22 |      BEGIN
   9 | 1|   26 |            C := 30;
  10 | 2|   30 |            E := C;
  11 | 2|   35 |            G := B * E;
  12 | 2|   43 |      END ELSE
  13 | 1|   43 |      BEGIN
  14 | 1|   46 |      B := B + 2;
  15 | 2|   53 |      D := B;
  16 | 2|   58 |      F := D * C;
  17 | 2|   66 |      END;
  18 | 1|   67 |H := C * B;
  19 | 1|   75 |I := B * C;
  20 | 1|   83 |WHILE H GTR C DO
  21 | 1|   93 |   BEGIN
  22 | 1|   93 |      C := C + 1;
  23 | 2|  100 |      F := 1 + 2;
  24 | 2|  106 |      G := G + 0;
  25 | 2|  113 |      A := A * 1;
  26 | 2|  120 |      B := B * 0;
  27 | 2|  127 |   END;
  28 | 1|  130 |END
  29 | 1|  130 |EOF
CODE FILE COPIED (130 WORDS)
CONSTANT TABLE COPIED (12 WORDS)
2 RECORDS WRITTEN INTO FILE 1
END OF COMPILATION FEBRUARY 12, 1975.  CLOCK TIME = 20:29:35.48.

29 CARDS WERE READ.
NO ERRORS WERE DETECTED.
```

```
***CODE SYNTHESIS FILTER***

BLOCK SELECTION METHOD WILL BE FIRST-IN-FIRST-OUT

***C O M P L E T E   T A B L E   D U M P***
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

***T A B L E   D U M P*** AT LOCATION = 0

CONSTANTS <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
CONSYM   |5   |10  |30  |2   |1   |0
CONTYPE  |INT |INT |INT |INT |INT |INT
CONVAL   |131 |132 |133 |134 |135 |136
CONINT   |5   |10  |30  |2   |1   |0

ADDR TABL <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

VALUE STACK (TOP AT 127)


CONT STK  <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
BLOCK#   |0
ENTRY    |0

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)


  ****  FORMATTED INTERMEDIATE CODE DUMP  ****
  LOC OFF    OP CODE    CNS ADR     RAW CODE      OPTIMIZATION
      SET                  /ETC                       TYPE
****************************************************************************
  0     0    |ENT       |C|131    |  00038083  |
  1     0    |TOGGLE    | |257    |  00080101  |
  2     0    |LOC       | |2      |  00020002  |
  3     0    |INT       | |1      |  00050001  |
  4     0    |STO       | |0      |  00280000  |
  5     0    |DEL       | |0      |  002A0000  |
  6     0    |LOC       | |3      |  00020003  |
  7     0    |INT       | |2      |  00050002  |
  8     0    |STO       | |0      |  00280000  |
  9     0    |DEL       | |0      |  002A0000  |
  10    0    |LOC       | |1      |  00020001  |
  11    0    |LOC       | |2      |  00020002  |
  12    0    |LOD       | |0      |  00270000  |
  13    0    |LOC       | |3      |  00020003  |
  14    0    |LOD       | |0      |  00270000  |
  15    0    |MUL       | |0      |  00140000  |
  16    0    |STO       | |0      |  00280000  |
  17    0    |DEL       | |0      |  002A0000  |
  18    0    |LOC       | |1      |  00020001  |
  19    0    |LOD       | |0      |  00270000  |
  20    0    |LOC       | |2      |  00020002  |
  21    0    |LOD       | |0      |  00270000  |
  22    0    |GTR       | |0      |  00210000  |
  23    0    |XIT       | |46     |  0004002E  |
  24    0    |XIT       | |26     |  0004001A  |
  25    0    |BSC       | |0      |  002F0000  |
  26    0    |ENT       | |1      |  00030001  |
  27    0    |LOC       | |3      |  00020003  |
  28    0    |INT       | |3      |  00050003  |
  29    0    |STO       | |0      |  00280000  |
  30    0    |DEL       | |0      |  002A0000  |
  31    0    |LOC       | |5      |  00020005  |
  32    0    |LOC       | |3      |  00020003  |
  33    0    |LOD       | |0      |  00270000  |
  34    0    |STO       | |0      |  00280000  |
  35    0    |DEL       | |0      |  002A0000  |
  36    0    |LOC       | |7      |  00020007  |
  37    0    |LOC       | |2      |  00020002  |
  38    0    |LOD       | |0      |  00270000  |
  39    0    |LOC       | |5      |  00020005  |
  40    0    |LOD       | |0      |  00270000  |
  41    0    |MUL       | |0      |  00140000  |
```

42	0	STO	0	00280000
43	0	DEL	0	002A0000
44	0	XIT	67	00040043
45	0	BRS	0	002E0000
46	0	ENT	1	00030001
47	0	LOC	2	00020002
48	0	LOC	2	00020002
49	0	LOD	0	00270000
50	0	INT	4	00050004
51	0	ADD	0	00100000
52	0	STO	0	00280000
53	0	DEL	0	002A0000
54	0	LOC	4	00020004
55	0	LOC	2	00020002
56	0	LOD	0	00270000
57	0	STO	0	00280000
58	0	DEL	0	002A0000
59	0	LOC	6	00020006
60	0	LOC	4	00020004
61	0	LOD	0	00270000
62	0	LOC	3	00020003
63	0	LOD	0	00270000
64	0	MUL	0	00140000
65	0	STO	0	00280000
66	0	DEL	0	002A0000
67	0	ENT	1	00030001
68	0	LOC	8	00020008
69	0	LOC	3	00020003
70	0	LOD	0	00270000
71	0	LOC	2	00020002
72	0	LOD	0	00270000
73	0	MUL	0	00140000
74	0	STO	0	00280000
75	0	DEL	0	002A0000
76	0	LOC	9	00020009
77	0	LOC	2	00020002
78	0	LOD	0	00270000
79	0	LOC	3	00020003
80	0	LOD	0	00270000
81	0	MUL	0	00140000
82	0	STO	0	00280000
83	0	DEL	0	002A0000
84	0	ENT	1	00030001
85	0	LOC	8	00020008
86	0	LOD	0	00270000
87	0	LOC	3	00020003
88	0	LOD	0	00270000
89	0	GTR	0	00210000
90	0	XIT	130	00040082
91	0	XIT	93	0004005D
92	0	BSC	0	002F0000
93	0	ENT	1	00030001
94	0	LOC	3	00020003
95	0	LOC	3	00020003
96	0	LOD	0	00270000
97	0	INT	5	00050005
98	0	ADD	0	00100000
99	0	STO	0	00280000
100	0	DEL	0	002A0000
101	0	LOC	6	00020006
102	0	INT	5	00050005
103	0	INT	4	00050004
104	0	ADD	0	00100000
105	0	STO	0	00280000
106	0	DEL	0	002A0000
107	0	LOC	7	00020007
108	0	LOC	7	00020007
109	0	LOD	0	00270000
110	0	INT	6	00050006
111	0	ADD	0	00100000
112	0	STO	0	00280000
113	0	DEL	0	002A0000
114	0	LOC	1	00020001
115	0	LOC	1	00020001
116	0	LOD	0	00270000
117	0	INT	5	00050005
118	0	MUL	0	00140000
119	0	STO	0	00280000
120	0	DEL	0	002A0000
121	0	LOC	2	00020002
122	0	LOC	2	00020002
123	0	LOD	0	00270000
124	0	INT	6	00050006
125	0	MUL	0	00140000
126	0	STO	0	00280000
127	0	DEL	0	002A0000
128	0	XIT	84	00040054
129	0	BRS	0	002E0000

```
130   0   |ENT          | |1     | 00030001  |
*******************************************************************************

      BASIC BLOCK #1
             BEGINNING AT 0, ENDING AT 0
             BLOCK TRAVERSED 0 TIMES
             0 REFERENCES:

***E N D    O F   C O M P L E T E   T A B L E   D U M P***

B+2     0   |LOC          | |2     | 00020002  |
B+3     0   |INT          | |1     | 00050001  |
B+4     0   |STO          | |0     | 00280000  |
B+5     0   |DEL          | |0     | 002A0000  |
B+6     0   |LOC          | |3     | 00020003  |
B+7     0   |INT          | |2     | 00050002  |
B+8     0   |STO          | |0     | 00280000  |
B+9     0   |DEL          | |0     | 002A0000  |
B+10    0   |LOC          | |1     | 00020001  |
B+11    0   |LOC          | |2     | 00020002  |
B+12    0   |LOD          | |0     | 00270000  |
B+13    0   |LOC          | |3     | 00020003  |
B+14    0   |LOD          | |0     | 00270000  |
B+15    0   |MUL          | |0     | 00140000  |
B+16    0   |STO          | |0     | 00280000  |
B+17    0   |DEL          | |0     | 002A0000  |
B+18    0   |LOC          | |1     | 00020001  |
B+19    0   |LOD          | |0     | 00270000  |
B+20    0   |LOC          | |2     | 00020002  |
B+21    0   |LOD          | |0     | 00270000  |
B+22    0   |GTR          | |0     | 00210000  |
B+23    0   |XIT          | |46    | 0004002E  |
B+24    0   |XIT          | |26    | 0004001A  |
B+25    0   |BSC          | |0     | 002F0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 2 IS:
 C(131:132,*,138)  C(138:131,*,140)  A(1,138,138)  A(3,132,132)  A(2,131,131)

 CONTROL AT 26, FROM BLK 1 TO BLK 2 (PASS 1)
C+26    0   |ENT          |C|135   | 00038087  |
B+27    0   |LOC          | |3     | 00020003  |
B+28    0   |INT          | |3     | 00050003  |
B+29    0   |STO          | |0     | 00280000  |
B+30    0   |DEL          | |0     | 002A0000  |
B+31    0   |LOC          | |5     | 00020005  |
B+32    0   |LOC          | |3     | 00020003  |
B+33    0   |LOD          | |0     | 00270000  |
B+34    0   |STO          | |0     | 00280000  |
B+35    0   |DEL          | |0     | 002A0000  |
B+36    0   |LOC          | |7     | 00020007  |
B+37    0   |LOC          | |2     | 00020002  |
B+38    0   |LOD          | |0     | 00270000  |
B+39    0   |LOC          | |5     | 00020005  |
B+40    0   |LOD          | |0     | 00270000  |
B+41    0   |MUL          | |0     | 00140000  |
B+42    0   |STO          | |0     | 00280000  |
B+43    0   |DEL          | |0     | 002A0000  |
B+44    0   |XIT          | |67    | 00040043  |
B+45    0   |BRS          | |0     | 002E0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 3 IS:
 C(138:131,*,140)  C(131:132,*,138)  C(131:133,*,142)  A(7,142,142)
 A(5,133,133)  A(3,133,133)  A(1,138,138)  A(2,131,131)

 CONTROL AT 67, FROM BLK 2 TO BLK 3 (PASS 1)
C+67    0   |ENT          |C|140   | 0003808C  |
B+68    0   |LOC          | |8     | 00020008  |
B+69    0   |LOC          | |3     | 00020003  |
B+70    0   |LOD          | |0     | 00270000  |
B+71    0   |LOC          | |2     | 00020002  |
B+72    0   |LOD          | |0     | 00270000  |
B+73    0   |MUL          | |0     | 00140000  |
B+74    0   |STO          | |0     | 00280000  |
B+75    0   |DEL          | |0     | 002A0000  |
B+76    0   |LOC          | |9     | 00020009  |
B+77    0   |LOC          | |2     | 00020002  |
B+78    0   |LOD          | |0     | 00270000  |
B+79    0   |LOC          | |3     | 00020003  |
B+80    0   |LOD          | |0     | 00270000  |
B+81    0   |MUL          | |0     | 00140000  |
B+82    0   |STO          | |0     | 00280000  |
B+83    0   |DEL          | |0     | 002A0000  |
B+84    0   |ENT          | |1     | 00030001  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 4 IS:
 C(131:133,*,142)  C(131:132,*,138)  C(138:131,*,140)  A(9,142,142)
 A(8,142,142)  A(7,142,142)  A(5,133,133)  A(3,133,133)  A(1,138,138)
```

```
A(2,131,131)

 CONTROL AT 84, FROM BLK 3 TO BLK 4 (PASS 1)
C+84    0   |ENT          |C|145    | 00038091 |
B+85    0   |LOC          | |8      | 00020008 |
B+86    0   |LOD          | |0      | 00270000 |
B+87    0   |LOC          | |3      | 00020003 |
B+88    0   |LOD          | |0      | 00270000 |
B+89    0   |GTR          | |0      | 00210000 |
B+90    0   |XIT          | |130    | 00040082 |
B+91    0   |XIT          | |93     | 0004005D |
B+92    0   |BSC          | |0      | 002F0000 |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 5 IS:
 C(138:131,*,140)  C(131:132,*,138)  C(131:133,*,142)  C(142:133,*,140)
 A(9,142,142)  A(8,142,142)  A(7,142,142)  A(5,133,133)  A(3,133,133)
 A(1,138,138)  A(2,131,131)

 CONTROL AT 93, FROM BLK 4 TO BLK 5 (PASS 1)
C+93    0   |ENT          |C|150    | 00038096 |
B+94    0   |LOC          | |3      | 00020003 |
B+95    0   |LOC          | |3      | 00020003 |
B+96    0   |LOD          | |0      | 00270000 |
B+97    0   |INT          | |5      | 00050005 |
B+98    0   |ADD          | |0      | 00100000 |
B+99    0   |STO          | |0      | 00280000 |
B+100   0   |DEL          | |0      | 002A0000 |
B+101   0   |LOC          | |6      | 00020006 |
B+102   0   |INT          | |5      | 00050005 |
B+103   0   |INT          | |4      | 00050004 |
B+104   0   |ADD          | |0      | 00100000 |
B+105   0   |STO          | |0      | 00280000 |
B+106   0   |DEL          | |0      | 002A0000 |
B+107   0   |LOC          | |7      | 00020007 |
B+108   0   |LOC          | |7      | 00020007 |
B+109   0   |LOD          | |0      | 00270000 |
B+110   0   |INT          | |6      | 00050006 |
B+111   0   |ADD          | |0      | 00100000 |
B+112   0   |STO          | |0      | 00280000 |
B+113   0   |DEL          | |0      | 002A0000 |
B+114   0   |LOC          | |1      | 00020001 |
B+115   0   |LOC          | |1      | 00020001 |
B+116   0   |LOD          | |0      | 00270000 |
B+117   0   |INT          | |5      | 00050005 |
B+118   0   |MUL          | |0      | 00140000 |
B+119   0   |STO          | |0      | 00280000 |
B+120   0   |DEL          | |0      | 002A0000 |
B+121   0   |LOC          | |2      | 00020002 |
B+122   0   |LOC          | |2      | 00020002 |
B+123   0   |LOD          | |0      | 00270000 |
B+124   0   |INT          | |6      | 00050006 |
B+125   0   |MUL          | |0      | 00140000 |
B+126   0   |STO          | |0      | 00280000 |
B+127   0   |DEL          | |0      | 002A0000 |
B+128   0   |XIT          | |84     | 00040054 |
B+129   0   |BRS          | |0      | 002E0000 |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 5 IS:
 C(142:133,*,140)  C(131:133,*,142)  C(131:132,*,138)  C(138:131,*,140)
 C(133:135,*,145)  C(131:133,*,145)  C(135:134,*,147)  C(142:136,*,142)
 C(138:135,*,138)  C(131:136,*,136)  A(2,136,136)  A(1,138,138)  A(7,142,142)
 A(6,147,147)  A(3,145,145)  A(9,142,142)  A(8,142,142)  A(5,133,133)

 CURRENT POOL FOR BLOCK# 4 IS:
 C(131:133,*,142)  C(131:132,*,138)  C(138:131,*,140)  A(9,142,142)
 A(8,142,142)  A(7,142,142)  A(5,133,133)  A(3,133,133)  A(1,138,138)
 A(2,131,131)
AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 4 IS:
 A(9,142,142)  A(8,142,142)  A(7,142,142)  A(5,133,133)  A(3,*,151)
 A(1,138,138)  A(2,*,152)

 CONTROL AT 84, FROM BLK 5 TO BLK 4 (PASS 2)
C+84    0   |ENT          |C|145    | 00038091 |
B+85    0   |LOC          | |8      | 00020008 |
B+86    0   |LOD          | |0      | 00270000 |
B+87    0   |LOC          | |3      | 00020003 |
B+88    0   |LOD          | |0      | 00270000 |
B+89    0   |GTR          |C|988    | 002183DC |    CONSTANT PROPAGATION
B+90    0   |XIT          | |130    | 00040082 |
B+91    0   |XIT          | |93     | 0004005D |
B+92    0   |BSC          | |0      | 002F0000 |
```

```
 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 6 IS:
 C(142:151,*,153)  A(9,142,142)  A(8,142,142)  A(7,142,142)  A(5,133,133)
 A(3,*,151)  A(1,138,138)  A(2,*,152)

BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 4 IS:
 C(142:151,*,153)  A(9,142,142)  A(8,142,142)  A(7,142,142)  A(5,133,133)
 A(3,*,151)  A(1,138,138)  A(2,*,152)

 CURRENT POOL FOR BLOCK# 5 IS:
 C(138:131,*,140)  C(131:132,*,138)  C(131:133,*,142)  C(142:133,*,140)
 A(9,142,142)  A(8,142,142)  A(7,142,142)  A(5,133,133)  A(3,133,133)
 A(1,138,138)  A(2,131,131)

AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 5 IS:
 A(9,142,142)  A(8,142,142)  A(7,142,142)  A(5,133,133)  A(3,*,154)
 A(1,138,138)  A(2,*,155)  C(142:154,*,156)


 CONTROL AT 130, FROM BLK 4 TO BLK 6 (PASS 1)
C+130  0    |ENT         |C|155    | 00038098  |
 CONTROL AT 93, FROM BLK 4 TO BLK 5 (PASS 2)
C+93    0   |ENT         |C|150    | 00038096  |
B+94    0   |LOC         | |3      | 00020003  |
B+95    0   |LOC         | |3      | 00020003  |
B+96    0   |LOD         | |0      | 00270000  |
B+97    0   |INT         | |5      | 00050005  |
B+98    0   |ADD         |C|976    | 001083D0  |   CONSTANT PROPAGATION
B+99    0   |STO         | |0      | 00280000  |
B+100   0   |DEL         | |0      | 002A0000  |
B+101   0   |LOC         | |6      | 00020006  |
B+102   0   |INT         | |5      | 00050005  |
B+103   0   |INT         | |4      | 00050004  |
B+104   0   |ADD         |C|972    | 001083CC  |   CONSTANT PROPAGATION
B+105   0   |STO         | |0      | 00280000  |
B+106   0   |DEL         | |0      | 002A0000  |
B+107   0   |LOC         | |7      | 00020007  |
B+108   0   |LOC         | |7      | 00020007  |
B+109   0   |LOD         | |0      | 00270000  |
B+110   0   |INT         | |6      | 00050006  |
B+111   0   |ADD         |C|968    | 001083C8  |   CONSTANT PROPAGATION
B+112   0   |STO         | |0      | 00280000  |
B+113   0   |DEL         | |0      | 002A0000  |
B+114   0   |LOC         | |1      | 00020001  |
B+115   0   |LOC         | |1      | 00020001  |
B+116   0   |LOD         | |0      | 00270000  |
B+117   0   |INT         | |5      | 00050005  |
B+118   0   |MUL         |C|964    | 001483C4  |   CONSTANT PROPAGATION
B+119   0   |STO         | |0      | 00280000  |
B+120   0   |DEL         | |0      | 002A0000  |
B+121   0   |LOC         | |2      | 00020002  |
B+122   0   |LOC         | |2      | 00020002  |
B+123   0   |LOD         | |0      | 00270000  |
B+124   0   |INT         | |6      | 00050006  |
B+125   0   |MUL         |C|960    | 001483C0  |   CONSTANT PROPAG  ION
B+126   0   |STO         | |0      | 00280000  |
B+127   0   |DEL         | |0      | 002A0000  |
B+128   0   |XIT         | |84     | 00040054  |
B+129   0   |BRS         | |0      | 002E0000  |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 5 IS:
 C(142:154,*,156)  C(154:135,*,157)  C(135:134,*,147)  C(142:136,*,142)
 C(138:135,*,138)  C(155:136,*,136)  A(2,136,136)  A(1,138,138)  A(7,142,142)
 A(6,147,147)  A(3,*,157)  A(9,142,142)  A(8,142,142)  A(5,133,133)

 CURRENT POOL FOR BLOCK# 4 IS:
 A(9,142,142)  A(8,142,142)  A(7,142,142)  A(5,133,133)  A(3,*,151)
 A(1,138,138)  A(2,*,152)

AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL


                         FINAL OPTIMIZATION RESULTS
#####################################################################
---------------------------------------------------------------------
```

```
OPTIMIZATION AT 15
          (131,132|15,138)     CONSTANT PROPAGATION

OPTIMIZATION AT 22
          (138,131|22,140)     CONSTANT PROPAGATION

OPTIMIZATION AT 41
          (131,133|41,142)     CONSTANT PROPAGATION

OPTIMIZATION AT 73
          (131,133|41,142)     COMM SUBEXP ELIM & CONS PROP

OPTIMIZATION AT 81
          (131,133|41,142)     COMM SUBEXP ELIM & CONS PROP

OPTIMIZATION AT 104
          (134,135|104,147)    CONSTANT PROPAGATION

OPTIMIZATION AT 111
          (136,142|111,142)    CONSTANT PROPAGATION

OPTIMIZATION AT 118
          (135,138|118,138)    CONSTANT PROPAGATION

OPTIMIZATION AT 125
          (136,155|125,136)    SIMPLIFICATION
```

**** FORMATTED INTERMEDIATE CODE DUMP ****

LOC	OFF SET	OP CODE	CNS	ADR /ETC	RAW CODE	OPTIMIZATION TYPE
0	0	ENT		131	00030083	
1	0	TOGGLE		257	00080101	
2	0	LOC		2	00020002	
3	0	INT		1	00050001	
4	0	STO		0	00280000	
5	0	DEL		0	002A0000	
6	0	LOC		3	00020003	
7	0	INT		2	00050002	
8	0	STO		0	00280000	
9	0	DEL		0	002A0000	
10	0	LOC		1	00020001	
11	0	LOC		2	00020002	
12	0	LOD		0	00270000	
13	0	LOC		3	00020003	
14	0	LOD		0	00270000	
15	0	MUL	C	1024	00148400	CONSTANT PROPAGATION
16	0	STO		0	00280000	
17	0	DEL		0	002A0000	
18	0	LOC		1	00020001	
19	0	LOD		0	00270000	
20	0	LOC		2	00020002	
21	0	LOD		0	00270000	
22	0	GTR	C	1020	002183FC	CONSTANT PROPAGATION
23	0	XIT		46	0004002E	
24	0	XIT		26	0004001A	
25	0	BSC		0	002F0000	
26	0	ENT		135	00030087	
27	0	LOC		3	00020003	
28	0	INT		3	00050003	
29	0	STO		0	00280000	
30	0	DEL		0	002A0000	
31	0	LOC		5	00020005	
32	0	LOC		3	00020003	
33	0	LOD		0	00270000	
34	0	STO		0	00280000	
35	0	DEL		0	002A0000	
36	0	LOC		7	00020007	
37	0	LOC		2	00020002	
38	0	LOD		0	00270000	
39	0	LOC		5	00020005	
40	0	LOD		0	00270000	
41	0	MUL	C	1012	001483F4	CONSTANT PROPAGATION
42	0	STO		0	00280000	
43	0	DEL		0	002A0000	
44	0	XIT		67	00040043	
45	0	BRS		0	002E0000	
46	0	ENT		1	00030001	
47	0	LOC		2	00020002	
48	0	LOC		2	00020002	
49	0	LOD		0	00270000	
50	0	INT		4	00050004	
51	0	ADD		0	00100000	
52	0	STO		0	00280000	
53	0	DEL		0	002A0000	
54	0	LOC		4	00020004	
55	0	LOC		2	00020002	

56	0	LOD		0	00270000	
57	0	STO		0	00280000	
58	0	DEL		0	002A0000	
59	0	LOC		6	00020006	
60	0	LOC		4	00020004	
61	0	LOD		0	00270000	
62	0	LOC		3	00020003	
63	0	LOD		0	00270000	
64	0	MUL		0	00140000	
65	0	STO		0	00280000	
66	0	DEL		0	002A0000	
67	0	ENT		140	0003008C	
68	0	LOC		8	00020008	
69	0	LOC		3	00020003	
70	0	LOD		0	00270000	
71	0	LOC		2	00020002	
72	0	LOD		0	00270000	
73	0	MUL	C	1002	001483EA	COMM SUBEXP ELIM & CONS PROP
74	0	STO		0	00280000	
75	0	DEL		0	002A0000	
76	0	LOC		9	00020009	
77	0	LOC		2	00020002	
78	0	LOD		0	00270000	
79	0	LOC		3	00020003	
80	0	LOD		0	00270000	
81	0	MUL	C	998	001483E6	COMM SUBEXP ELIM & CONS PROP
82	0	STO		0	00280000	
83	0	DEL		0	002A0000	
84	0	ENT		145	00030091	
85	0	LOC		8	00020008	
86	0	LOD		0	00270000	
87	0	LOC		3	00020003	
88	0	LOD		0	00270000	
89	0	GTR		988	002103DC	
90	0	XIT		130	00040082	
91	0	XIT		93	0004005D	
92	0	BSC		0	002F0000	
93	0	ENT		150	00030096	
94	0	LOC		3	00020003	
95	0	LOC		3	00020003	
96	0	LOD		0	00270000	
97	0	INT		5	00050005	
98	0	ADD		976	001003D0	
99	0	STO		0	00280000	
100	0	DEL		0	002A0000	
101	0	LOC		6	00020006	
102	0	INT		5	00050005	
103	0	INT		4	00050004	
104	0	ADD	C	972	001083CC	CONSTANT PROPAGATION
105	0	STO		0	00280000	
106	0	DEL		0	002A0000	
107	0	LOC		7	00020007	
108	0	LOC		7	00020007	
109	0	LOD		0	00270000	
110	0	INT		6	00050006	
111	0	ADD	C	968	001083C8	CONSTANT PROPAGATION
112	0	STO		0	00280000	
113	0	DEL		0	002A0000	
114	0	LOC		1	00020001	
115	0	LOC		1	00020001	
116	0	LOD		0	00270000	
117	0	INT		5	00050005	
118	0	MUL	C	964	001483C4	CONSTANT PROPAGATION
119	0	STO		0	00280000	
120	0	DEL		0	002A0000	
121	0	LOC		2	00020002	
122	0	LOC		2	00020002	
123	0	LOD		0	00270000	
124	0	INT		6	00050006	
125	0	MUL	C	960	001483C0	SIMPLIFICATION
126	0	STO		0	00280000	
127	0	DEL		0	002A0000	
128	0	XIT		84	00040054	
129	0	BRS		0	002E0000	
130	0	ENT		155	0003009B	

****

FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

FINAL POOL FOR BLOCK# 1 IS:
(NULL)

```
FINAL POOL FOR  BLOCK# 2 IS:
C(131:132,*,138)  C(138:131,*,140)  A(1,138,138)  A(3,132,132)  A(2,131,131)


FINAL POOL FOR  BLOCK# 3 IS:
C(138:131,*,140)  C(131:132,*,138)  C(131:133,*,142)  A(7,142,142)
A(5,133,133)  A(3,133,133)  A(1,138,138)  A(2,131,131)


FINAL POOL FOR  BLOCK# 4 IS:
A(9,142,142)  A(8,142,142)  A(7,142,142)  A(5,133,133)  A(3,*,160)
A(1,138,138)  A(2,*,161)


FINAL POOL FOR  BLOCK# 5 IS:
A(9,142,142)  A(8,142,142)  A(7,142,142)  A(5,133,133)  A(3,*,154)
A(1,138,138)  A(2,*,155)  C(142:154,*,156)


FINAL POOL FOR  BLOCK# 6 IS:
C(142:151,*,153)  A(9,142,142)  A(8,142,142)  A(7,142,142)  A(5,133,133)
A(3,*,151)  A(1,138,138)  A(2,*,152)


     BASIC BLOCK #1
           BEGINNING AT 0, ENDING AT 25
           BLOCK TRAVERSED 1 TIMES
           0 REFERENCES:
     BASIC BLOCK #2
           BEGINNING AT 26, ENDING AT 45
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
              1
     BASIC BLOCK #3
           BEGINNING AT 67, ENDING AT 84
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
              2
     BASIC BLOCK #4
           BEGINNING AT 84, ENDING AT 92
           BLOCK TRAVERSED 2 TIMES
           1 REFERENCES:
              3
     BASIC BLOCK #5
           BEGINNING AT 93, ENDING AT 129
           BLOCK TRAVERSED 2 TIMES
           1 REFERENCES:
              4
     BASIC BLOCK #6
           BEGINNING AT 130, ENDING AT 130
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
              4

BLOCK SUMMARY DATA:

BLOCK NUMBER                    NUMBER OF PASSES

     1                                1
     2                                1
     3                                1
     4                                2
     5                                2
     6                                1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 8
USING THE FIRST-IN-FIRST-OUT  BLOCK SELECTION ALGORITHM.

TIME FOR THE OPTIMIZATION WAS 0.270 SECONDS.

                 ***CODE SYNTHESIS FILTER IS COMPLETE***
```

```
CARD |BL| SYL |
   1 | 0|   0 |COMMENT     TEST PROGRAM # 5          ;
$EXECUTE
   3 | 0|   1 |BEGIN LOCAL A, B, C, D, E, F, G, H, I;
   4 | 1|   1 |B := 5;
   5 | 1|   5 |C := 10;
   6 | 1|   9 |A := B * C;
   7 | 1|  17 |IF A GTR B THEN
   8 | 1|  22 |BEGIN
   9 | 1|  26 |        E := C;
  10 | 2|  31 |        G := B * E;
  11 | 2|  39 |END
  12 | 2|  39 |ELSE
  13 | 1|  39 |BEGIN
  14 | 1|  42 |        D := B;
  15 | 2|  47 |        F := D * C;
  16 | 2|  55 |END;
  17 | 1|  56 |H := C * B;
  18 | 1|  64 |IF A GTR B THEN
  19 | 1|  69 |BEGIN
  20 | 1|  73 |        C := 30;
  21 | 2|  77 |        E := C;
  22 | 2|  82 |        G := B * C;
  23 | 2|  90 |END
  24 | 2|  90 |ELSE
  25 | 1|  90 |BEGIN
  26 | 1|  93 |        B := 20;
  27 | 2|  97 |        D := B;
  28 | 2| 102 |        F := D * C;
  29 | 2| 110 |END;
  30 | 1| 111 |H := C * B;
  31 | 1| 119 |I := B * C;
  32 | 1| 127 |END
  33 | 1| 127 |EOF EOF
CODE FILE COPIED (127 WORDS)
CONSTANT TABLE COPIED (8 WORDS)
2 RECORDS WRITTEN INTO FILE 1
END OF COMPILATION FEBRUARY 12, 1975.  CLOCK TIME = 20:31:47.40.

33 CARDS WERE READ.
NO ERRORS WERE DETECTED.
```

```
                         ***CODE SYNTHESIS FILTER***

BLOCK SELECTION METHOD WILL BE LAST-IN-FIRST-OUT

***C O M P L E T E   T A B L E   D U M P***
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

***T A B L E   D U M P*** AT LOCATION = 0

CONSTANTS <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
CONSYM    |5   |10  |30  |20
CCNTYPE   |INT |INT |INT |INT
CONVAL    |128 |129 |130 |131
CONINT    |5   |10  |30  |20

ADDR TABL <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

VALUE STACK (TOP AT 127)

CONT STK  <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
BLOCK#    |0
ENTRY     |0

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

 ****  FORMATTED INTERMEDIATE CODE DUMP  ****
 LOC OFF    OP CODE     CNS ADR    RAW CODE       OPTIMIZATION
     SET                    /ETC                       TYPE
*******************************************************************************
 0      0   |ENT        |C|128    | 00038080 |
 1      0   |TOGGLE     | |257    | 000B0101 |
 2      0   |LOC        | |2      | 00020002 |
 3      0   |INT        | |1      | 00050001 |
 4      0   |STO        | |0      | 00280000 |
 5      0   |DEL        | |0      | 002A0000 |
 6      0   |LOC        | |3      | 00020003 |
 7      0   |INT        | |2      | 00050002 |
 8      0   |STO        | |0      | 00280000 |
 9      0   |DEL        | |0      | 002A0000 |
 10     0   |LOC        | |1      | 00020001 |
 11     0   |LOC        | |2      | 00020002 |
 12     0   |LOD        | |0      | 00270000 |
 13     0   |LOC        | |3      | 00020003 |
 14     0   |LOD        | |0      | 00270000 |
 15     0   |MUL        | |0      | 00140000 |
 16     0   |STO        | |0      | 00280000 |
 17     0   |DEL        | |0      | 002A0000 |
 18     0   |LOC        | |1      | 00020001 |
 19     0   |LOD        | |0      | 00270000 |
 20     0   |LOC        | |2      | 00020002 |
 21     0   |LOD        | |0      | 00270000 |
 22     0   |GTR        | |0      | 00210000 |
 23     0   |XIT        | |42     | 0004002A |
 24     0   |XIT        | |26     | 0004001A |
 25     0   |BSC        | |0      | 002F0000 |
 26     0   |ENT        | |1      | 00030001 |
 27     0   |LOC        | |5      | 00020005 |
 28     0   |LOC        | |3      | 00020003 |
 29     0   |LOD        | |0      | 00270000 |
 30     0   |STO        | |0      | 00280000 |
 31     0   |DEL        | |0      | 002A0000 |
 32     0   |LOC        | |7      | 00020007 |
 33     0   |LOC        | |2      | 00020002 |
 34     0   |LOD        | |0      | 00270000 |
 35     0   |LOC        | |5      | 00020005 |
 36     0   |LOD        | |0      | 00270000 |
 37     0   |MUL        | |0      | 00140000 |
 38     0   |STO        | |0      | 00280000 |
 39     0   |DEL        | |0      | 002A0000 |
 40     0   |XIT        | |56     | 00040038 |
 41     0   |BRS        | |0      | 002E0000 |
```

42	0	ENT		1	00030001
43	0	LOC		4	00020004
44	0	LOC		2	00020002
45	0	LOD		0	00270000
46	0	STO		0	00280000
47	0	DEL		0	002A0000
48	0	LOC		6	00020006
49	0	LOC		4	00020004
50	0	LOD		0	00270000
51	0	LOC		3	00020003
52	0	LOD		0	00270000
53	0	MUL		0	00140000
54	0	STO		0	00280000
55	0	DEL		0	002A0000
56	0	ENT		1	00030001
57	0	LOC		8	00020008
58	0	LOC		3	00020003
59	0	LOD		0	00270000
60	0	LOC		2	00020002
61	0	LOD		0	00270000
62	0	MUL		0	00140000
63	0	STO		0	00280000
64	0	DEL		0	002A0000
65	0	LOC		1	00020001
66	0	LOD		0	00270000
67	0	LOC		2	00020002
68	0	LOD		0	00270000
69	0	GTR		0	00210000
70	0	XIT		93	0004005D
71	0	XIT		73	00040049
72	0	BSC		0	002F0000
73	0	ENT		1	00030001
74	0	LOC		3	00020003
75	0	INT		3	00050003
76	0	STO		0	00280000
77	0	DEL		0	002A0000
78	0	LOC		5	00020005
79	0	LOC		3	00020003
80	0	LOD		0	00270000
81	0	STO		0	00280000
82	0	DEL		0	002A0000
83	0	LOC		7	00020007
84	0	LOC		2	00020002
85	0	LOD		0	00270000
86	0	LOC		3	00020003
87	0	LOD		0	00270000
88	0	MUL		0	00140000
89	0	STO		0	00280000
90	0	DEL		0	002A0000
91	0	XIT		111	0004006F
92	0	BRS		0	002E0000
93	0	ENT		1	00030001
94	0	LOC		2	00020002
95	0	INT		4	00050004
96	0	STO		0	00280000
97	0	DEL		0	002A0000
98	0	LOC		4	00020004
99	0	LOC		2	00020002
100	0	LOD		0	00270000
101	0	STO		0	00280000
102	0	DEL		0	002A0000
103	0	LOC		6	00020006
104	0	LOC		4	00020004
105	0	LOD		0	00270000
106	0	LOC		3	00020003
107	0	LOD		0	00270000
108	0	MUL		0	00140000
109	0	STO		0	00280000
110	0	DEL		0	002A0000
111	0	ENT		1	00030001
112	0	LOC		8	00020008
113	0	LOC		3	00020003
114	0	LOD		0	00270000
115	0	LOC		2	00020002
116	0	LOD		0	00270000
117	0	MUL		0	00140000
118	0	STO		0	00280000
119	0	DEL		0	002A0000
120	0	LOC		9	00020009
121	0	LOC		2	00020002
122	0	LOD		0	00270000
123	0	LOC		3	00020003
124	0	LOD		0	00270000
125	0	MUL		0	00140000
126	0	STO		0	00280000
127	0	DEL		0	002A0000

```
BASIC BLOCK #1
      BEGINNING AT 0, ENDING AT 0
      BLOCK TRAVERSED 0 TIMES
      0 REFERENCES:

***E N D    O F    C O M P L E T E    T A B L E    D U M P***

B+2     0   |LOC          |  |2         |  00020002  |
B+3     0   |INT          |  |1         |  00050001  |
B+4     0   |STO          |  |0         |  00280000  |
B+5     0   |DEL          |  |0         |  002A0000  |
B+6     0   |LOC          |  |3         |  00020003  |
B+7     0   |INT          |  |2         |  00050002  |
B+8     0   |STO          |  |0         |  00280000  |
B+9     0   |DEL          |  |0         |  002A0000  |
B+10    0   |LOC          |  |1         |  00020001  |
B+11    0   |LOC          |  |2         |  00020002  |
B+12    0   |LOD          |  |0         |  00270000  |
B+13    0   |LOC          |  |3         |  00020003  |
B+14    0   |LOD          |  |0         |  00270000  |
B+15    0   |MUL          |  |0         |  00140000  |
B+16    0   |STO          |  |0         |  00280000  |
B+17    0   |DEL          |  |0         |  002A0000  |
B+18    0   |LOC          |  |1         |  00020001  |
B+19    0   |LOD          |  |0         |  00270000  |
B+20    0   |LOC          |  |2         |  00020002  |
B+21    0   |LOD          |  |0         |  00270000  |
B+22    0   |GTR          |  |0         |  00210000  |
B+23    0   |XIT          |  |42        |  0004002A  |
B+24    0   |XIT          |  |26        |  0004001A  |
B+25    0   |BSC          |  |0         |  002F0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 2 IS:
 C(128:129,*,133)  C(133:128,*,135)  A(1,133,133)  A(3,129,129)  A(2,128,128)

 CONTROL AT 26, FROM BLK 1 TO BLK 2 (PASS 1)
C+26    0   |ENT          |C |132       |  00038084  |
B+27    0   |LOC          |  |5         |  00020005  |
B+28    0   |LOC          |  |3         |  00020003  |
B+29    0   |LOD          |  |0         |  00270000  |
B+30    0   |STO          |  |0         |  00280000  |
B+31    0   |DEL          |  |0         |  002A0000  |
B+32    0   |LOC          |  |7         |  00020007  |
B+33    0   |LOC          |  |2         |  00020002  |
B+34    0   |LOD          |  |0         |  00270000  |
B+35    0   |LOC          |  |5         |  00020005  |
B+36    0   |LOD          |  |0         |  00270000  |
B+37    0   |MUL          |  |0         |  00140000  |
B+38    0   |STO          |  |0         |  00280000  |
B+39    0   |DEL          |  |0         |  002A0000  |
B+40    0   |XIT          |  |56        |  00040038  |
B+41    0   |BRS          |  |0         |  002E0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 3 IS:
 C(133:128,*,135)  C(128:129,*,133)  A(7,133,133)  A(5,129,129)  A(1,133,133)
 A(3,129,129)  A(2,128,128)

 CONTROL AT 56, FROM BLK 2 TO BLK 3 (PASS 1)
C+56    0   |ENT          |C |137       |  00038089  |
B+57    0   |LOC          |  |8         |  00020008  |
B+58    0   |LOC          |  |3         |  00020003  |
B+59    0   |LOD          |  |0         |  00270000  |
B+60    0   |LOC          |  |2         |  00020002  |
B+61    0   |LOD          |  |0         |  00270000  |
B+62    0   |MUL          |  |0         |  00140000  |
B+63    0   |STO          |  |0         |  00280000  |
B+64    0   |DEL          |  |0         |  002A0000  |
B+65    0   |LOC          |  |1         |  00020001  |
B+66    0   |LOD          |  |0         |  00270000  |
B+67    0   |LOC          |  |2         |  00020002  |
B+68    0   |LOD          |  |0         |  00270000  |
B+69    0   |GTR          |  |0         |  00210000  |
B+70    0   |XIT          |  |93        |  0004005D  |
B+71    0   |XIT          |  |73        |  00040049  |
B+72    0   |BSC          |  |0         |  002F0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 4 IS:
 C(128:129,*,133)  C(133:128,*,135)  A(8,133,133)  A(7,133,133)  A(5,129,129)
 A(1,133,133)  A(3,129,129)  A(2,128,128)

 CONTROL AT 73, FROM BLK 3 TO BLK 4 (PASS 1)
C+73    0   |ENT          |C |142       |  0003808E  |
B+74    0   |LOC          |  |3         |  00020003  |
B+75    0   |INT          |  |3         |  00050003  |
B+76    0   |STO          |  |0         |  00280000  |
B+77    0   |DEL          |  |0         |  002A0000  |
B+78    0   |LOC          |  |5         |  00020005  |
```

```
B+79   0   LOC               3          00020003
B+80   0   LOD               0          00270000
B+81   0   STO               0          00280000
B+82   0   DEL               0          002A0000
B+83   0   LOC               7          00020007
B+84   0   LOC               2          00020002
B+85   0   LOD               0          00270000
B+86   0   LOC               3          00020003
B+87   0   LOD               0          00270000
B+88   0   MUL               0          00140000
B+89   0   STO               0          00280000
B+90   0   DEL               0          002A0000
B+91   0   XIT               111        0004006F
B+92   0   BRS               0          002E0000
```

```
INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 5 IS:
C(133:128,*,135)  C(128:129,*,133)  C(128:130,*,137)  A(7,137,137)
A(5,130,130)  A(3,130,130)  A(8,133,133)  A(1,133,133)  A(2,128,128)
```

```
CONTROL AT 111, FROM BLK 4 TO BLK 5 (PASS 1)
C+111  0   ENT           C   147        00038093
B+112  0   LOC               8          00020008
B+113  0   LOC               3          00020003
B+114  0   LOD               0          00270000
B+115  0   LOC               2          00020002
B+116  0   LOD               0          00270000
B+117  0   MUL               0          00140000
B+118  0   STO               0          00280000
B+119  0   DEL               0          002A0000
B+120  0   LOC               9          00020009
B+121  0   LOC               2          00020002
B+122  0   LOD               0          00270000
B+123  0   LOC               3          00020003
B+124  0   LOD               0          00270000
B+125  0   MUL               0          00140000
B+126  0   STO               0          00280000
B+127  0   DEL               0          002A0000
```

```
                              FINAL OPTIMIZATION RESULTS
##########################################################################################
------------------------------------------------------------------------------------------

OPTIMIZATION AT 15
          (128,129|15,133)     CONSTANT PROPAGATION

OPTIMIZATION AT 22
          (133,128|22,135)     CONSTANT PROPAGATION

OPTIMIZATION AT 37
          (128,129|15,133)     COMM SUBEXP ELIM & CONS PROP

OPTIMIZATION AT 62
          (128,129|15,133)     COMM SUBEXP ELIM & CONS PROP

OPTIMIZATION AT 69
          (133,128|22,135)     COMM SUBEXP ELIM & CONS PROP

OPTIMIZATION AT 88
          (130,128|88,137)     CONSTANT PROPAGATION

OPTIMIZATION AT 117
          (130,128|88,137)     COMM SUBEXP ELIM & CONS PROP

OPTIMIZATION AT 125
          (130,128|88,137)     COMM SUBEXP ELIM & CONS PROP
```

```
 ****  FORMATTED INTERMEDIATE CODE DUMP  ****
 LOC OFF    OP CODE      CNS ADR    RAW CODE      OPTIMIZATION
     SET                  /ETC                        TYPE
*********************************************************************************
 0     0   ENT               128        00030080
 1     0   TOGGLE            257        00030101
 2     0   LOC               2          00020002
 3     0   INT               1          00050001
 4     0   STO               0          00280000
 5     0   DEL               0          002A0000
 6     0   LOC               3          00020003
 7     0   INT               2          00050002
 8     0   STO               0          00280000
 9     0   DEL               0          002A0000
 10    0   LOC               1          00020001
 11    0   LOC               2          00020002
 12    0   LOD               0          00270000
 13    0   LOC               3          00020003
 14    0   LOD               0          00270000
```

15	0	MUL	C	1024	00148400	CONSTANT PROPAGATION
16	0	STO		0	00280000	
17	0	DEL		0	002A0000	
18	0	LOC		1	00020001	
19	0	LOD		0	00270000	
20	0	LOC		2	00020002	
21	0	LOD		0	00270000	
22	0	GTR	C	1020	002183FC	CONSTANT PROPAGATION
23	0	XIT		42	0004002A	
24	0	XIT		26	0004001A	
25	0	BSC		0	002F0000	
26	0	ENT		132	00030084	
27	0	LOC		5	00020005	
28	0	LOC		3	00020003	
29	0	LOD		0	00270000	
30	0	STO		0	00280000	
31	0	DEL		0	002A0000	
32	0	LOC		7	00020007	
33	0	LOC		2	00020002	
34	0	LOD		0	00270000	
35	0	LOC		5	00020005	
36	0	LOD		0	00270000	
37	0	MUL	C	1012	001483F4	COMM SUBEXP ELIM & CONS PROP
38	0	STO		0	00280000	
39	0	DEL		0	002A0000	
40	0	XIT		56	00040038	
41	0	BRS		0	002E0000	
42	0	ENT		1	00030001	
43	0	LOC		4	00020004	
44	0	LOC		2	00020002	
45	0	LOD		0	00270000	
46	0	STO		0	00280000	
47	0	DEL		0	002A0000	
48	0	LOC		6	00020006	
49	0	LOC		4	00020004	
50	0	LOD		0	00270000	
51	0	LOC		3	00020003	
52	0	LOD		0	00270000	
53	0	MUL		0	00140000	
54	0	STO		0	00280000	
55	0	DEL		0	002A0000	
56	0	ENT		137	00030089	
57	0	LOC		8	00020008	
58	0	LOC		3	00020003	
59	0	LOD		0	00270000	
60	0	LOC		2	00020002	
61	0	LOD		0	00270000	
62	0	MUL	C	1004	001483EC	COMM SUBEXP ELIM & CONS PROP
63	0	STO		0	00280000	
64	0	DEL		0	002A0000	
65	0	LOC		1	00020001	
66	0	LOD		0	00270000	
67	0	LOC		2	00020002	
68	0	LOD		0	00270000	
69	0	GTR	C	1000	002183E8	COMM SUBEXP ELIM & CONS PROP
70	0	XIT		93	0004005D	
71	0	XIT		73	00040049	
72	0	BSC		0	002F0000	
73	0	ENT		142	0003008E	
74	0	LOC		3	00020003	
75	0	INT		3	00050003	
76	0	STO		0	00280000	
77	0	DEL		0	002A0000	
78	0	LOC		5	00020005	
79	0	LOC		3	00020003	
80	0	LOD		0	00270000	
81	0	STO		0	00280000	
82	0	DEL		0	002A0000	
83	0	LOC		7	00020007	
84	0	LOC		2	00020002	
85	0	LOD		0	00270000	
86	0	LOC		3	00020003	
87	0	LOD		0	00270000	
88	0	MUL	C	992	001483E0	CONSTANT PROPAGATION
89	0	STO		0	00280000	
90	0	DEL		0	002A0000	
91	0	XIT		111	0004006F	
92	0	BRS		0	002E0000	
93	0	ENT		1	00030001	
94	0	LOC		2	00020002	
95	0	INT		4	00050004	
96	0	STO		0	00280000	
97	0	DEL		0	002A0000	
98	0	LOC		4	00020004	
99	0	LOC		2	00020002	
100	0	LOD		0	00270000	
101	0	STO		0	00280000	
102	0	DEL		0	002A0000	

```
103  0  LOC     6      00020006
104  0  LOC     4      00020004
105  0  LOD     0      00270000
106  0  LOC     3      00020003
107  0  LOD     0      00270000
108  0  MUL     0      00140000
109  0  STO     0      00280000
110  0  DEL     0      002A0000
111  0  ENT     147    00030093
112  0  LOC     8      00020008
113  0  LOC     3      00020003
114  0  LOD     0      00270000
115  0  LOC     2      00020002
116  0  LOD     0      00270000
117  0  MUL   C 982    001483D6   COMM SUBEXP ELIM & CONS PROP
118  0  STO     0      00280000
119  0  DEL     0      002A0000
120  0  LOC     9      00020009
121  0  LOC     2      00020002
122  0  LOD     0      00270000
123  0  LOC     3      00020003
124  0  LOD     0      00270000
125  0  MUL   C 978    001483D2   COMM SUBEXP ELIM & CONS PROP
126  0  STO     0      00280000
127  0  DEL     0      002A0000
************************************************************************************


      FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

 FINAL POCL FOR  BLOCK# 1 IS:
(NULL)


 FINAL POCL FOR  BLOCK# 2 IS:
 C(128:129,*,133)  C(133:128,*,135)  A(1,133,133)  A(3,129,129)  A(2,128,128)


 FINAL POCL FOR  BLOCK# 3 IS:
 C(133:128,*,135)  C(128:129,*,133)  A(7,133,133)  A(5,129,129)  A(1,133,133)
 A(3,129,129)  A(2,128,128)


 FINAL POCL FOR  BLOCK# 4 IS:
 C(128:129,*,133)  C(133:128,*,135)  A(8,133,133)  A(7,133,133)  A(5,129,129)
 A(1,133,133)  A(3,129,129)  A(2,128,128)


 FINAL POOL FOR  BLOCK# 5 IS:
 C(133:128,*,135)  C(128:129,*,133)  C(128:130,*,137)  A(7,137,137)
 A(5,130,130)  A(3,130,130)  A(8,133,133)  A(1,133,133)  A(2,128,128)


     BASIC BLOCK #1
           BEGINNING AT 0, ENDING AT 25
           BLOCK TRAVERSED 1 TIMES
           0 REFERENCES:
     BASIC BLOCK #2
           BEGINNING AT 26, ENDING AT 41
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
              1
     BASIC BLOCK #3
           BEGINNING AT 56, ENDING AT 72
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
              2
     BASIC BLOCK #4
           BEGINNING AT 73, ENDING AT 92
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
              3
     BASIC BLOCK #5
           BEGINNING AT 111, ENDING AT 127
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
              4

BLOCK SUMMARY DATA:
```

BLOCK NUMBER	NUMBER OF PASSES
1	1
2	1
3	1
4	1
5	1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 5
USING THE LAST-IN-FIRST-OUT BLOCK SELECTION ALGORITHM.

TIME FOR THE OPTIMIZATION WAS 0.170 SECONDS.

CODE SYNTHESIS FILTER IS COMPLETE

```
CARD |BL|  SYL |
  1  | 0|    0 |COMMENT     TEST PROGRAM # 6        ;
$EXECUTE
  3  | 0|    1 |BEGIN LOCAL A,B,C,D,E,F,G,H,I,J;
  4  | 1|    1 |READ(A, B);
  5  | 1|    7 |WHILE A GTR B DO
  6  | 1|   17 |BEGIN
  7  | 1|   17 |     IF A GTR B THEN
  8  | 2|   22 |     BEGIN
  9  | 2|   26 |          IF A GTR B THEN
 10  | 3|   31 |          BEGIN
 11  | 3|   35 |               B := 10;
 12  | 4|   39 |               A := B * C;
 13  | 4|   47 |          END
 14  | 4|   47 |          ELSE
 15  | 3|   47 |          BEGIN
 16  | 3|   50 |               B := 20;
 17  | 4|   54 |               D := B;
 18  | 4|   59 |               F := D * C;
 19  | 4|   67 |          END;
 20  | 3|   68 |     E := B;
 21  | 3|   73 |     H := E * C;
 22  | 3|   81 |END
 23  | 3|   81 |ELSE
 24  | 2|   81 |BEGIN
 25  | 2|   84 |     B := 30;
 26  | 3|   88 |     J := C * B;
 27  | 3|   96 |END;
 28  | 2|   97 |E := C;
 29  | 2|  102 |I := B * C;
 30  | 2|  110 |END;
 31  | 1|  113 |END
 32  | 1|  113 |EOF
CODE FILE COPIED (113 WORDS)
CONSTANT TABLE COPIED (6 WORDS)
2 RECORDS WRITTEN INTO FILE 1
END OF COMPILATION FEBRUARY 12, 1975.  CLOCK TIME = 20:42:29.46.

32 CARDS WERE READ.
NO ERRORS WERE DETECTED.
```

```
***CODE SYNTHESIS FILTER***

BLOCK SELECTION METHOD WILL BE LAST-IN-FIRST-OUT

***C O M P L E T E   T A B L E   D U M P***
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

***T A B L E   D U M P*** AT LOCATION = 0

CONSTANTS <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
CONSYM   |10  |20  |30
CCNTYPE  |INT |INT |INT
CONVAL   |114 |115 |116
CONINT   |10  |20  |30

ADDR TABL <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

VALUE STACK (TOP AT 127)


CONT STK  <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
BLOCK#   |0
ENTRY    |0

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)


 ****  FORMATTED INTERMEDIATE CODE DUMP  ****
 LOC OFF     OP CODE     CNS ADR     RAW CODE      OPTIMIZATION
     SET                     /ETC                      TYPE
*******************************************************************************
 0     0    |ENT        |C|114   |  00038072  |
 1     0    |TOGGLE     | |257   |  00080101  |
 2     0    |LOC        | |1     |  00020001  |
 3     0    |INT        | |0     |  00050000  |
 4     0    |STD        | |0     |  00290000  |
 5     0    |LOC        | |2     |  00020002  |
 6     0    |INT        | |0     |  00050000  |
 7     0    |STD        | |0     |  00290000  |
 8     0    |ENT        | |1     |  00030001  |
 9     0    |LOC        | |1     |  00020001  |
 10    0    |LOD        | |0     |  00270000  |
 11    0    |LOC        | |2     |  00020002  |
 12    0    |LOD        | |0     |  00270000  |
 13    0    |GTR        | |0     |  00210000  |
 14    0    |XIT        | |113   |  00040071  |
 15    0    |XIT        | |17    |  00040011  |
 16    0    |BSC        | |0     |  002F0000  |
 17    0    |ENT        | |1     |  00030001  |
 18    0    |LOC        | |1     |  00020001  |
 19    0    |LOD        | |0     |  00270000  |
 20    0    |LOC        | |2     |  00020002  |
 21    0    |LOD        | |0     |  00270000  |
 22    0    |GTR        | |0     |  00210000  |
 23    0    |XIT        | |84    |  00040054  |
 24    0    |XIT        | |26    |  0004001A  |
 25    0    |BSC        | |0     |  002F0000  |
 26    0    |ENT        | |1     |  00030001  |
 27    0    |LOC        | |1     |  00020001  |
 28    0    |LOD        | |0     |  00270000  |
 29    0    |LOC        | |2     |  00020002  |
 30    0    |LOD        | |0     |  00270000  |
 31    0    |GTR        | |0     |  00210000  |
 32    0    |XIT        | |50    |  00040032  |
 33    0    |XIT        | |35    |  00040023  |
 34    0    |BSC        | |0     |  002F0000  |
 35    0    |ENT        | |1     |  00030001  |
 36    0    |LOC        | |2     |  00020002  |
 37    0    |INT        | |1     |  00050001  |
 38    0    |STO        | |0     |  00280000  |
 39    0    |DEL        | |0     |  002A0000  |
 40    0    |LOC        | |1     |  00020001  |
 41    0    |LOC        | |2     |  00020002  |
```

42	0	LOD	0	00270000
43	0	LOC	3	00020003
44	0	LOD	0	00270000
45	0	MUL	0	00140000
46	0	STO	0	00280000
47	0	DEL	0	002A0000
48	0	XIT	68	00040044
49	0	BRS	0	002E0000
50	0	ENT	1	00030001
51	0	LOC	2	00020002
52	0	INT	2	00050002
53	0	STO	0	00280000
54	0	DEL	0	002A0000
55	0	LOC	4	00020004
56	0	LOC	2	00020002
57	0	LOD	0	00270000
58	0	STO	0	00280000
59	0	DEL	0	002A0000
60	0	LOC	6	00020006
61	0	LOC	4	00020004
62	0	LOD	0	00270000
63	0	LOC	3	00020003
64	0	LOD	0	00270000
65	0	MUL	0	00140000
66	0	STO	0	00280000
67	0	DEL	0	002A0000
68	0	ENT	1	00030001
69	0	LOC	5	00020005
70	0	LOC	2	00020002
71	0	LOD	0	00270000
72	0	STO	0	00280000
73	0	DEL	0	002A0000
74	0	LOC	8	00020008
75	0	LOC	5	00020005
76	0	LOD	0	00270000
77	0	LOC	3	00020003
78	0	LOD	0	00270000
79	0	MUL	0	00140000
80	0	STO	0	00280000
81	0	DEL	0	002A0000
82	0	XIT	97	00040061
83	0	BRS	0	002E0000
84	0	ENT	1	00030001
85	0	LOC	2	00020002
86	0	INT	3	00050003
87	0	STO	0	00280000
88	0	DEL	0	002A0000
89	0	LOC	10	0002000A
90	0	LOC	3	00020003
91	0	LOD	0	00270000
92	0	LOC	2	00020002
93	0	LOD	0	00270000
94	0	MUL	0	00140000
95	0	STO	0	00280000
96	0	DEL	0	002A0000
97	0	ENT	1	00030001
98	0	LOC	5	00020005
99	0	LOC	3	00020003
100	0	LOD	0	00270000
101	0	STO	0	00280000
102	0	DEL	0	002A0000
103	0	LOC	9	00020009
104	0	LOC	2	00020002
105	0	LOD	0	00270000
106	0	LOC	3	00020003
107	0	LOD	0	00270000
108	0	MUL	0	00140000
109	0	STO	0	00280000
110	0	DEL	0	002A0000
111	0	XIT	8	00040008
112	0	BRS	0	002E0000
113	0	ENT	1	00030001

```
**************************************************************************************

      BASIC BLOCK #1
            BEGINNING AT 0, ENDING AT 0
            BLOCK TRAVERSED 0 TIMES
            0 REFERENCES:

***E N D   O F   C O M P L E T E   T A B L E   D U M P***
```

B+2	0	LOC	1	00020001
B+3	0	INT	0	00050000
B+4	0	STO	0	00290000
B+5	0	LOC	2	00020002
B+6	0	INT	0	00050000
B+7	0	STO	0	00290000

```
B+8     0    |ENT           | 11        | 00030001  |
 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 2 IS:
 A(2,*,118)  A(1,*,117)

 CONTROL AT 8, FROM BLK 1 TO BLK 2 (PASS 1)
C+8     0    |ENT           |C|118      | 00038076  |
B+9     0    |LOC           | |1        | 00020001  |
B+10    0    |LOD           | |0        | 00270000  |
B+11    0    |LOC           | |2        | 00020002  |
B+12    0    |LOD           | |0        | 00270000  |
B+13    0    |GTR           | |0        | 00210000  |
B+14    0    |XIT           | |113      | 00040071  |
B+15    0    |XIT           | |17       | 00040011  |
B+16    0    |BSC           | |0        | 002F0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 3 IS:
 C(117:118,*,119)  A(2,*,118)  A(1,*,117)


 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 4 IS:
 C(117:118,*,119)  A(2,*,118)  A(1,*,117)

 CONTROL AT 17, FROM BLK 2 TO BLK 4 (PASS 1)
C+17    0    |ENT           |C|128      | 00038080  |
B+18    0    |LOC           | |1        | 00020001  |
B+19    0    |LOD           | |0        | 00270000  |
B+20    0    |LOC           | |2        | 00020002  |
B+21    0    |LOD           | |0        | 00270000  |
B+22    0    |GTR           | |0        | 00210000  |
B+23    0    |XIT           | |84       | 00040054  |
B+24    0    |XIT           | |26       | 0004001A  |
B+25    0    |BSC           | |0        | 002F0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 5 IS:
 C(117:118,*,119)  A(2,*,118)  A(1,*,117)


 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 6 IS:
 C(117:118,*,119)  A(2,*,118)  A(1,*,117)

 CONTROL AT 26, FROM BLK 4 TO BLK 6 (PASS 1)
C+26    0    |ENT           |C|138      | 0003808A  |
B+27    0    |LOC           | |1        | 00020001  |
B+28    0    |LOD           | |0        | 00270000  |
B+29    0    |LOC           | |2        | 00020002  |
B+30    0    |LOD           | |0        | 00270000  |
B+31    0    |GTR           | |0        | 00210000  |
B+32    0    |XIT           | |50       | 00040032  |
B+33    0    |XIT           | |35       | 00040023  |
B+34    0    |BSC           | |0        | 002F0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 7 IS:
 C(117:118,*,119)  A(2,*,118)  A(1,*,117)


 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 8 IS:
 C(117:118,*,119)  A(2,*,118)  A(1,*,117)

 CONTROL AT 35, FROM BLK 6 TO BLK 8 (PASS 1)
C+35    0    |ENT           |C|148      | 00038094  |
B+36    0    |LOC           | |2        | 00020002  |
B+37    0    |INT           | |1        | 00050001  |
B+38    0    |STO           | |0        | 00280000  |
B+39    0    |DEL           | |0        | 002A0000  |
B+40    0    |LOC           | |1        | 00020001  |
B+41    0    |LOC           | |2        | 00020002  |
B+42    0    |LOD           | |0        | 00270000  |
B+43    0    |LOC           | |3        | 00020003  |
B+44    0    |LOD           | |0        | 00270000  |
B+45    0    |MUL           | |0        | 00140000  |
B+46    0    |STO           | |0        | 00280000  |
B+47    0    |DEL           | |0        | 002A0000  |
B+48    0    |XIT           | |68       | 00040044  |
B+49    0    |BRS           | |0        | 002E0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 9 IS:
 C(117:118,*,119)  C(114:120,*,121)  A(1,*,121)  A(3,*,120)  A(2,114,114)

 CONTROL AT 68, FROM BLK 8 TO BLK 9 (PASS 1)
C+68    0    |ENT           |C|153      | 00038099  |
B+69    0    |LOC           | |5        | 00020005  |
B+70    0    |LOC           | |2        | 00020002  |
B+71    0    |LOD           | |0        | 00270000  |
B+72    0    |STO           | |0        | 00280000  |
B+73    0    |DEL           | |0        | 002A0000  |
B+74    0    |LOC           | |8        | 00020008  |
B+75    0    |LOC           | |5        | 00020005  |
```

```
B+76   0   |LOD     | |0    | 00270000 |
B+77   0   |LOC     | |3    | 00020003 |
B+78   0   |LOD     | |0    | 00270000 |
B+79   0   |MUL     | |0    | 00140000 |
B+80   0   |STO     | |0    | 00280000 |
B+81   0   |DEL     | |0    | 002A0000 |
B+82   0   |XIT     | |97   | 00040061 |
B+83   0   |BRS     | |0    | 002E0000 |

 INITIAL CURRENT OPTIMIZED PCOL TO BE USED FOR  BLOCK# 10 IS:
 C(114:120,*,121)  C(117:118,*,119)  A(8,*,121)  A(5,114,114)  A(1,*,121)
 A(3,*,120)  A(2,114,114)

 CONTROL AT 97, FROM BLK 9 TO BLK 10 (PASS 1)
C+97   0   |ENT     |C|158  | 0003809E |
B+98   0   |LOC     | |5    | 00020005 |
B+99   0   |LOC     | |3    | 00020003 |
B+100  0   |LOD     | |0    | 00270000 |
B+101  0   |STO     | |0    | 00280000 |
B+102  0   |DEL     | |0    | 002A0000 |
B+103  0   |LOC     | |9    | 00020009 |
B+104  0   |LOC     | |2    | 00020002 |
B+105  0   |LOD     | |0    | 00270000 |
B+106  0   |LOC     | |3    | 00020003 |
B+107  0   |LOD     | |0    | 00270000 |
B+108  0   |MUL     | |0    | 00140000 |
B+109  0   |STO     | |0    | 00280000 |
B+110  0   |DEL     | |0    | 002A0000 |
B+111  0   |XIT     | |8    | 00040008 |
B+112  0   |BRS     | |0    | 002E0000 |
BEFORE THE MEET OPERATION:

 OUTPUT PCOL FROM BLOCK# 10 IS:
 C(117:118,*,119)  C(114:120,*,121)  A(9,*,121)  A(5,*,120)  A(8,*,121)
 A(1,*,121)  A(3,*,120)  A(2,114,114)


 CURRENT POOL FOR BLOCK# 2 IS:
 A(2,*,118)  A(1,*,117)

AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POCL IS THE SAME AS OLD CURRENT POOL

 CONTROL AT 50, FROM BLK 6 TO BLK 7 (PASS 1)
C+50   0   |ENT     |C|143  | 0003808F |
B+51   0   |LOC     | |2    | 00020002 |
B+52   0   |INT     | |2    | 00050002 |
B+53   0   |STO     | |0    | 00280000 |
B+54   0   |DEL     | |0    | 002A0000 |
B+55   0   |LOC     | |4    | 00020004 |
B+56   0   |LOC     | |2    | 00020002 |
B+57   0   |LOD     | |0    | 00270000 |
B+58   0   |STO     | |0    | 00280000 |
B+59   0   |DEL     | |0    | 002A0000 |
B+60   0   |LOC     | |6    | 00020006 |
B+61   0   |LOC     | |4    | 00020004 |
B+62   0   |LOD     | |0    | 00270000 |
B+63   0   |LOC     | |3    | 00020003 |
B+64   0   |LOD     | |0    | 00270000 |
B+65   0   |MUL     | |0    | 00140000 |
B+66   0   |STO     | |0    | 00280000 |
B+67   0   |DEL     | |0    | 002A0000 |
B+68   0   |ENT     |C|153  | 00038099 |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 7 IS:
 C(117:118,*,119)  C(115:124,*,125)  A(3,*,124)  A(6,*,125)  A(4,115,115)
 A(2,115,115)  A(1,*,117)


 CURRENT POOL FOR BLOCK# 9 IS:
 C(117:118,*,119)  C(114:120,*,121)  A(1,*,121)  A(3,*,120)  A(2,114,114)
AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 9 IS:
 A(1,*,126)  A(3,*,127)  A(2,*,128)  C(128:127,*,129)


 CONTROL AT 68, FROM BLK 7 TO BLK 9 (PASS 2)
C+68   0   |ENT     |C|153  | 00038099 |
B+69   0   |LOC     | |5    | 00020005 |
B+70   0   |LOC     | |2    | 00020002 |
B+71   0   |LOD     | |0    | 00270000 |
B+72   0   |STO     | |0    | 00280000 |
B+73   0   |DEL     | |0    | 002A0000 |
```

```
B+74   0  |LOC      | |8      | 00020008 |
B+75   0  |LOC      | |5      | 00020005 |
B+76   0  |LOD      | |0      | 00270000 |
B+77   0  |LOC      | |3      | 00020003 |
B+78   0  |LOD      | |0      | 00270000 |
B+79   0  |MUL      |C|992    | 001483E0 |    COMM SUBEXP ELIM
B+80   0  |STO      | |0      | 00280000 |
B+81   0  |DEL      | |0      | 002A0000 |
B+82   0  |XIT      | |97     | 00040061 |
B+83   0  |BRS      | |0      | 002E0000 |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 9 IS:
 C(128:127,*,129)  A(8,*,129)  A(5,*,128)  A(1,*,126)  A(3,*,127)  A(2,*,128)


 CURRENT POOL FOR BLOCK# 10 IS:
 C(114:120,*,121)  C(117:118,*,119)  A(8,*,121)  A(5,114,114)  A(1,*,121)
 A(3,*,120)  A(2,114,114)

AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 10 IS:
 A(8,*,130)  A(5,*,131)  A(1,*,132)  A(3,*,133)  A(2,*,131)


 CONTROL AT 97, FROM BLK 9 TO BLK 10 (PASS 2)
C+97   0  |ENT      |C|158    | 0003809E |
B+98   0  |LOC      | |5      | 00020005 |
B+99   0  |LOC      | |3      | 00020003 |
B+100  0  |LOD      | |0      | 00270000 |
B+101  0  |STO      | |0      | 00280000 |
B+102  0  |DEL      | |0      | 002A0000 |
B+103  0  |LOC      | |9      | 00020009 |
B+104  0  |LOC      | |2      | 00020002 |
B+105  0  |LOD      | |0      | 00270000 |
B+106  0  |LOC      | |3      | 00020003 |
B+107  0  |LOD      | |0      | 00270000 |
B+108  0  |MUL      |C|984    | 001483D8 |    COMM SUBEXP ELIM
B+109  0  |STO      | |0      | 00280000 |
B+110  0  |DEL      | |0      | 002A0000 |
B+111  0  |XIT      | |8      | 00040008 |
B+112  0  |BRS      | |0      | 002E0000 |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 10 IS:
 C(131:133,*,134)  A(9,*,134)  A(5,*,133)  A(8,*,130)  A(1,*,132)  A(3,*,133)
 A(2,*,131)


 CURRENT POOL FOR BLOCK# 2 IS:
 A(2,*,122)  A(1,*,123)

AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

 CONTROL AT 84, FROM BLK 4 TO BLK 5 (PASS 1)
C+84   0  |ENT      |C|133    | 00038085 |
B+85   0  |LOC      | |2      | 00020002 |
B+86   0  |INT      | |3      | 00050003 |
B+87   0  |STO      | |0      | 00280000 |
B+88   0  |DEL      | |0      | 002A0000 |
B+89   0  |LOC      | |10     | 0002000A |
B+90   0  |LOC      | |3      | 00020003 |
B+91   0  |LOD      | |0      | 00270000 |
B+92   0  |LOC      | |2      | 00020002 |
B+93   0  |LOD      | |0      | 00270000 |
B+94   0  |MUL      | |0      | 00140000 |
B+95   0  |STO      | |0      | 00280000 |
B+96   0  |DEL      | |0      | 002A0000 |
B+97   0  |ENT      |C|158    | 0003809E |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 5 IS:
 C(117:118,*,119)  C(137:116,*,138)  A(3,*,137)  A(10,*,138)  A(2,116,116)
 A(1,*,117)


 CURRENT POOL FOR BLOCK# 10 IS:
 A(8,*,130)  A(5,*,131)  A(1,*,132)  A(3,*,133)  A(2,*,131)

AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 10 IS:
 A(1,*,139)  A(3,*,140)  A(2,*,141)
```

```
 CONTROL AT 97, FROM BLK 5 TO BLK 10 (PASS 3)
C+97    0    |ENT          |C|158     | 0003809E  |
B+98    0    |LOC          | |5       | 00020005  |
B+99    0    |LOC          | |3       | 00020003  |
B+100   0    |LOD          | |0       | 00270000  |
B+101   0    |STO          | |0       | 00280000  |
B+102   0    |DEL          | |0       | 002A0000  |
B+103   0    |LOC          | |9       | 00020009  |
B+104   0    |LOC          | |2       | 00020002  |
B+105   0    |LOD          | |0       | 00270000  |
B+106   0    |LOC          | |3       | 00020003  |
B+107   0    |LOD          | |0       | 00270000  |
B+108   0    |MUL          |C|984     | 001483D8  |
B+109   0    |STO          | |0       | 00280000  |
B+110   0    |DEL          | |0       | 002A0000  |
B+111   0    |XIT          | |8       | 00040008  |
B+112   0    |BRS          | |0       | 002E0000  |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 10 IS:
 C(141:140,*,142)  A(9,*,142)  A(5,*,140)  A(1,*,139)  A(3,*,140)  A(2,*,141)


 CURRENT POOL FOR BLOCK# 2 IS:
 A(2,*,135)  A(1,*,136)

AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

 CONTROL AT 113, FROM BLK 2 TO BLK 3 (PASS 1)
C+113   0    |ENT          |C|123     | 0003807B  |

                         FINAL OPTIMIZATION RESULTS
################################################################################
--------------------------------------------------------------------------------

OPTIMIZATION AT 22
         (117,118|13,119)     COMM SUBEXP ELIM

OPTIMIZATION AT 31
         (117,118|13,119)     COMM SUBEXP ELIM

OPTIMIZATION AT 79
         (127,128|45,129)     COMM SUBEXP ELIM


 ****  FORMATTED INTERMEDIATE CODE DUMP  ****
 LOC OFF     OP CODE     CNS ADR     RAW CODE        OPTIMIZATION
     SET                     /ETC                         TYPE
******************************************************************************
 0    0      |ENT          | |114     | 00030072  |
 1    0      |TOGGLE       | |257     | 00080101  |
 2    0      |LOC          | |1       | 00020001  |
 3    0      |INT          | |0       | 00050000  |
 4    0      |STD          | |0       | 00290000  |
 5    0      |LOC          | |2       | 00020002  |
 6    0      |INT          | |0       | 00050000  |
 7    0      |STD          | |0       | 00290000  |
 8    0      |ENT          | |118     | 00030076  |
 9    0      |LOC          | |1       | 00020001  |
 10   0      |LOD          | |0       | 00270000  |
 11   0      |LOC          | |2       | 00020002  |
 12   0      |LOD          | |0       | 00270000  |
 13   0      |GTR          | |1024    | 00210400  |
 14   0      |XIT          | |113     | 00040071  |
 15   0      |XIT          | |17      | 00040011  |
 16   0      |BSC          | |0       | 002F0000  |
 17   0      |ENT          | |128     | 00030080  |
 18   0      |LOC          | |1       | 00020001  |
 19   0      |LOD          | |0       | 00270000  |
 20   0      |LOC          | |2       | 00020002  |
 21   0      |LOD          | |0       | 00270000  |
 22   0      |GTR          |C|1016    | 002183F8  |   COMM SUBEXP ELIM
 23   0      |XIT          | |84      | 00040054  |
 24   0      |XIT          | |26      | 0004001A  |
 25   0      |BSC          | |0       | 002F0000  |
 26   0      |ENT          | |138     | 0003008A  |
 27   0      |LOC          | |1       | 00020001  |
 28   0      |LOD          | |0       | 00270000  |
 29   0      |LOC          | |2       | 00020002  |
 30   0      |LOD          | |0       | 00270000  |
 31   0      |GTR          |C|1008    | 002183F0  |   COMM SUBEXP ELIM
 32   0      |XIT          | |50      | 00040032  |
```

33	0	XIT		35	00040023	
34	0	BSC		0	002F0000	
35	0	ENT		148	00030094	
36	0	LOC		2	00020002	
37	0	INT		1	00050001	
38	0	STO		0	00280000	
39	0	DEL		0	002A0000	
40	0	LOC		1	00020001	
41	0	LOC		2	00020002	
42	0	LOD		0	00270000	
43	0	LOC		3	00020003	
44	0	LOD		0	00270000	
45	0	MUL		1000	001403E8	
46	0	STO		0	00280000	
47	0	DEL		0	002A0000	
48	0	XIT		68	00040044	
49	0	BRS		0	002E0000	
50	0	ENT		143	0003008F	
51	0	LOC		2	00020002	
52	0	INT		2	00050002	
53	0	STO		0	00280000	
54	0	DEL		0	002A0000	
55	0	LOC		4	00020004	
56	0	LOC		2	00020002	
57	0	LOD		0	00270000	
58	0	STO		0	00280000	
59	0	DEL		0	002A0000	
60	0	LOC		6	00020006	
61	0	LOC		4	00020004	
62	0	LOD		0	00270000	
63	0	LOC		3	00020003	
64	0	LOD		0	00270000	
65	0	MUL		980	001403D4	
66	0	STO		0	00280000	
67	0	DEL		0	002A0000	
68	0	ENT		153	00030099	
69	0	LOC		5	00020005	
70	0	LOC		2	00020002	
71	0	LOD		0	00270000	
72	0	STO		0	00280000	
73	0	DEL		0	002A0000	
74	0	LOC		8	00020008	
75	0	LOC		5	00020005	
76	0	LOD		0	00270000	
77	0	LOC		3	00020003	
78	0	LOD		0	00270000	
79	0	MUL	C	992	001483E0	COMM SUBEXP ELIM
80	0	STO		0	00280000	
81	0	DEL		0	002A0000	
82	0	XIT		97	00040061	
83	0	BRS		0	002E0000	
84	0	ENT		133	00030085	
85	0	LOC		2	00020002	
86	0	INT		3	00050003	
87	0	STO		0	00280000	
88	0	DEL		0	002A0000	
89	0	LOC		10	0002000A	
90	0	LOC		3	00020003	
91	0	LOD		0	00270000	
92	0	LOC		2	00020002	
93	0	LOD		0	00270000	
94	0	MUL		976	001403D0	
95	0	STO		0	00280000	
96	0	DEL		0	002A0000	
97	0	ENT		158	0003009E	
98	0	LOC		5	00020005	
99	0	LOC		3	00020003	
100	0	LOD		0	00270000	
101	0	STO		0	00280000	
102	0	DEL		0	002A0000	
103	0	LOC		9	00020009	
104	0	LOC		2	00020002	
105	0	LOD		0	00270000	
106	0	LOC		3	00020003	
107	0	LOD		0	00270000	
108	0	MUL		984	001403D8	
109	0	STO		0	00280000	
110	0	DEL		0	002A0000	
111	0	XIT		8	00040008	
112	0	BRS		0	002E0000	
113	0	ENT		123	0003007B	

****

FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

```
FINAL POCL FOR  BLOCK# 1 IS:
(NULL)


FINAL POCL FOR  BLOCK# 2 IS:
A(2,*,143)  A(1,*,144)


FINAL POCL FOR  BLOCK# 3 IS:
C(117:118,*,119)  A(2,*,118)  A(1,*,117)


FINAL POCL FOR  BLOCK# 4 IS:
C(117:118,*,119)  A(2,*,118)  A(1,*,117)


FINAL POOL FOR  BLOCK# 5 IS:
C(117:118,*,119)  A(2,*,118)  A(1,*,117)


FINAL POCL FOR  BLOCK# 6 IS:
C(117:118,*,119)  A(2,*,118)  A(1,*,117)


FINAL POCL FOR  BLOCK# 7 IS:
C(117:118,*,119)  A(2,*,118)  A(1,*,117)


FINAL POCL FOR  BLOCK# 8 IS:
C(117:118,*,119)  A(2,*,118)  A(1,*,117)


FINAL POCL FOR  BLOCK# 9 IS:
A(1,*,126)  A(3,*,127)  A(2,*,128)  C(128:127,*,129)


FINAL POCL FOR  BLOCK# 10 IS:
A(1,*,139)  A(3,*,140)  A(2,*,141)


     BASIC BLOCK #1
           BEGINNING AT 0, ENDING AT 8
           BLOCK TRAVERSED 1 TIMES
           0 REFERENCES:
     BASIC BLOCK #2
           BEGINNING AT 8, ENDING AT 16
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
               1
     BASIC BLOCK #3
           BEGINNING AT 113, ENDING AT 113
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
               2
     BASIC BLOCK #4
           BEGINNING AT 17, ENDING AT 25
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
               2
     BASIC BLOCK #5
           BEGINNING AT 84, ENDING AT 97
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
               4
     BASIC BLOCK #6
           BEGINNING AT 26, ENDING AT 34
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
               4
     BASIC BLOCK #7
           BEGINNING AT 50, ENDING AT 68
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
               6
     BASIC BLOCK #8
           BEGINNING AT 35, ENDING AT 49
```

```
          BLOCK TRAVERSED 1 TIMES
          1 REFERENCES:
               6
     BASIC BLOCK #9
          BEGINNING AT 68, ENDING AT 83
          BLOCK TRAVERSED 2 TIMES
          1 REFERENCES:
               8
     BASIC BLOCK #10
          BEGINNING AT 97, ENDING AT 112
          BLOCK TRAVERSED 3 TIMES
          1 REFERENCES:
               9

BLOCK SUMMARY DATA:

BLOCK NUMBER                      NUMBER OF PASSES

     1                                  1
     2                                  1
     3                                  1
     4                                  1
     5                                  1
     6                                  1
     7                                  1
     8                                  1
     9                                  2
     10                                 3

TOTAL NUMBER OF BLOCKS PROCESSED WAS 13
USING THE LAST-IN-FIRST-OUT  BLOCK SELECTION ALGORITHM.

TIME FOR THE OPTIMIZATION WAS 0.333 SECONDS.

                  ***CODE SYNTHESIS FILTER IS COMPLETE***
```

```
***CODE SYNTHESIS FILTER***

BLOCK SELECTION METHOD WILL BE STEEPEST DESCENT (MINIMUM CURRENT POOL)

***C O M P L E T E   T A B L E   D U M P***
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

***T A B L E   D U M P*** AT LOCATION = 0

CONSTANTS <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
CONSYM    |10  |20  |30
CCNTYPE   |INT |INT |INT
CONVAL    |114 |115 |116
CONINT    |10  |20  |30

ADDR TABL <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

VALUE STACK (TOP AT 127)


CONT STK  <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
BLOCK#    |0
ENTRY     |0

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)


 ****  FORMATTED INTERMEDIATE CODE DUMP  ****
 LOC OFF     OP CODE     CNS ADR      RAW CODE       OPTIMIZATION
      SET                    /ETC                         TYPE
**************************************************************************
 0     0     |ENT        |C|114     |  00038072  |
 1     0     |TOGGLE     | |257     |  00080101  |
 2     0     |LOC        | |1       |  00020001  |
 3     0     |INT        | |0       |  00050000  |
 4     0     |STD        | |0       |  00290000  |
 5     0     |LOC        | |2       |  00020002  |
 6     0     |INT        | |0       |  00050000  |
 7     0     |STD        | |0       |  00290000  |
 8     0     |ENT        | |1       |  00030001  |
 9     0     |LOC        | |1       |  00020001  |
 10    0     |LOD        | |0       |  00270000  |
 11    0     |LOC        | |2       |  00020002  |
 12    0     |LOD        | |0       |  00270000  |
 13    0     |GTR        | |0       |  00210000  |
 14    0     |XIT        | |113     |  00040071  |
 15    0     |XIT        | |17      |  00040011  |
 16    0     |BSC        | |0       |  002F0000  |
 17    0     |ENT        | |1       |  00030001  |
 18    0     |LOC        | |1       |  00020001  |
 19    0     |LOD        | |0       |  00270000  |
 20    0     |LOC        | |2       |  00020002  |
 21    0     |LOD        | |0       |  00270000  |
 22    0     |GTR        | |0       |  00210000  |
 23    0     |XIT        | |84      |  00040054  |
 24    0     |XIT        | |26      |  0004001A  |
 25    0     |BSC        | |0       |  002F0000  |
 26    0     |ENT        | |1       |  00030001  |
 27    0     |LOC        | |1       |  00020001  |
 28    0     |LOD        | |0       |  00270000  |
 29    0     |LOC        | |2       |  00020002  |
 30    0     |LOD        | |0       |  00270000  |
 31    0     |GTR        | |0       |  00210000  |
 32    0     |XIT        | |50      |  00040032  |
 33    0     |XIT        | |35      |  00040023  |
 34    0     |BSC        | |0       |  002F0000  |
 35    0     |ENT        | |1       |  00030001  |
 36    0     |LOC        | |2       |  00020002  |
 37    0     |INT        | |1       |  00050001  |
 38    0     |STO        | |0       |  00280000  |
 39    0     |DEL        | |0       |  002A0000  |
 40    0     |LOC        | |1       |  00020001  |
 41    0     |LOC        | |2       |  00020002  |
```

42	0	LOD	0	00270000
43	0	LOC	3	00020003
44	0	LOD	0	00270000
45	0	MUL	0	00140000
46	0	STO	0	00280000
47	0	DEL	0	002A0000
48	0	XIT	68	00040044
49	0	BRS	0	002E0000
50	0	ENT	1	00030001
51	0	LOC	2	00020002
52	0	INT	2	00050002
53	0	STO	0	00280000
54	0	DEL	0	002A0000
55	0	LOC	4	00020004
56	0	LOC	2	00020002
57	0	LOD	0	00270000
58	0	STO	0	00280000
59	0	DEL	0	002A0000
60	0	LOC	6	00020006
61	0	LOC	4	00020004
62	0	LOD	0	00270000
63	0	LOC	3	00020003
64	0	LOD	0	00270000
65	0	MUL	0	00140000
66	0	STO	0	00280000
67	0	DEL	0	002A0000
68	0	ENT	1	00030001
69	0	LOC	5	00020005
70	0	LOC	2	00020002
71	0	LOD	0	00270000
72	0	STO	0	00280000
73	0	DEL	0	002A0000
74	0	LOC	8	00020008
75	0	LOC	5	00020005
76	0	LOD	0	00270000
77	0	LOC	3	00020003
78	0	LOD	0	00270000
79	0	MUL	0	00140000
80	0	STO	0	00280000
81	0	DEL	0	002A0000
82	0	XIT	97	00040061
83	0	BRS	0	002E0000
84	0	ENT	1	00030001
85	0	LOC	2	00020002
86	0	INT	3	00050003
87	0	STO	0	00280000
88	0	DEL	0	002A0000
89	0	LOC	10	0002000A
90	0	LOC	3	00020003
91	0	LOD	0	00270000
92	0	LOC	2	00020002
93	0	LOD	0	00270000
94	0	MUL	0	00140000
95	0	STO	0	00280000
96	0	DEL	0	002A0000
97	0	ENT	1	00030001
98	0	LOC	5	00020005
99	0	LOC	3	00020003
100	0	LOD	0	00270000
101	0	STO	0	00280000
102	0	DEL	0	002A0000
103	0	LOC	9	00020009
104	0	LOC	2	00020002
105	0	LOD	0	00270000
106	0	LOC	3	00020003
107	0	LOD	0	00270000
108	0	MUL	0	00140000
109	0	STO	0	00280000
110	0	DEL	0	002A0000
111	0	XIT	8	00040008
112	0	BRS	0	002E0000
113	0	ENT	1	00030001

```
******************************************************************************

      BASIC BLOCK #1
            BEGINNING AT 0, ENDING AT 0
            BLOCK TRAVERSED 0 TIMES
            0 REFERENCES:

***E N D   O F   C O M P L E T E   T A B L E   D U M P***
```

B+2	0	LOC	1	00020001
B+3	0	INT	0	00050000
B+4	0	STD	0	00290000
B+5	0	LOC	2	00020002
B+6	0	INT	0	00050000
B+7	0	STD	0	00290000

```
B+8    0   |ENT            | 11      | 00030001  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 2 IS:
 A(2,*,118)  A(1,*,117)

 CONTROL AT 8, FROM BLK 1 TO BLK 2 (PASS 1)
C+8     0   |ENT          |C|118      | 00038076  |
B+9     0   |LOC          | |1        | 00020001  |
B+10    0   |LOD          | |0        | 00270000  |
B+11    0   |LOC          | |2        | 00020002  |
B+12    0   |LOD          | |0        | 00270000  |
B+13    0   |GTR          | |0        | 00210000  |
B+14    0   |XIT          | |113      | 00040071  |
B+15    0   |XIT          | |17       | 00040011  |
B+16    0   |BSC          | |0        | 002F0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 3 IS:
 C(117:118,*,119)  A(2,*,118)  A(1,*,117)

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 4 IS:
 C(117:118,*,119)  A(2,*,118)  A(1,*,117)

 CONTROL AT 17, FROM BLK 2 TO BLK 4 (PASS 1)
C+17    0   |ENT          |C|128      | 00038080  |
B+18    0   |LOC          | |1        | 00020001  |
B+19    0   |LOD          | |0        | 00270000  |
B+20    0   |LOC          | |2        | 00020002  |
B+21    0   |LOD          | |0        | 00270000  |
B+22    0   |GTR          | |0        | 00210000  |
B+23    0   |XIT          | |84       | 00040054  |
B+24    0   |XIT          | |26       | 0004001A  |
B+25    0   |BSC          | |0        | 002F0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 5 IS:
 C(117:118,*,119)  A(2,*,118)  A(1,*,117)

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 6 IS:
 C(117:118,*,119)  A(2,*,118)  A(1,*,117)

 CONTROL AT 26, FROM BLK 4 TO BLK 6 (PASS 1)
C+26    0   |ENT          |C|138      | 0003808A  |
B+27    0   |LOC          | |1        | 00020001  |
B+28    0   |LOD          | |0        | 00270000  |
B+29    0   |LOC          | |2        | 00020002  |
B+30    0   |LOD          | |0        | 00270000  |
B+31    0   |GTR          | |0        | 00210000  |
B+32    0   |XIT          | |50       | 00040032  |
B+33    0   |XIT          | |35       | 00040023  |
B+34    0   |BSC          | |0        | 002F0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 7 IS:
 C(117:118,*,119)  A(2,*,118)  A(1,*,117)

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 8 IS:
 C(117:118,*,119)  A(2,*,118)  A(1,*,117)

 CONTROL AT 35, FROM BLK 6 TO BLK 8 (PASS 1)
C+35    0   |ENT          |C|148      | 00038094  |
B+36    0   |LOC          | |2        | 00020002  |
B+37    0   |INT          | |1        | 00050001  |
B+38    0   |STO          | |0        | 00280000  |
B+39    0   |DEL          | |0        | 002A0000  |
B+40    0   |LOC          | |1        | 00020001  |
B+41    0   |LOC          | |2        | 00020002  |
B+42    0   |LOD          | |0        | 00270000  |
B+43    0   |LOC          | |3        | 00020003  |
B+44    0   |LOD          | |0        | 00270000  |
B+45    0   |MUL          | |0        | 00140000  |
B+46    0   |STO          | |0        | 00280000  |
B+47    0   |DEL          | |0        | 002A0000  |
B+48    0   |XIT          | |68       | 00040044  |
B+49    0   |BRS          | |0        | 002E0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 9 IS:
 C(117:118,*,119)  C(114:120,*,121)  A(1,*,121)  A(3,*,120)  A(2,114,114)

 CONTROL AT 50, FROM BLK 6 TO BLK 7 (PASS 1)
C+50    0   |ENT          |C|143      | 0003808F  |
B+51    0   |LOC          | |2        | 00020002  |
B+52    0   |INT          | |2        | 00050002  |
B+53    0   |STO          | |0        | 00280000  |
B+54    0   |DEL          | |0        | 002A0000  |
B+55    0   |LOC          | |4        | 00020004  |
B+56    0   |LOC          | |2        | 00020002  |
B+57    0   |LOD          | |0        | 00270000  |
```

```
B+58   0   |STO          | |0       | 00280000 |
B+59   0   |DEL          | |0       | 002A0000 |
B+60   0   |LOC          | |6       | 00020006 |
B+61   0   |LOC          | |4       | 00020004 |
B+62   0   |LOD          | |0       | 00270000 |
B+63   0   |LOC          | |3       | 00020003 |
B+64   0   |LOD          | |0       | 00270000 |
B+65   0   |MUL          | |0       | 00140000 |
B+66   0   |STO          | |0       | 00280000 |
B+67   0   |DEL          | |0       | 002A0000 |
B+68   0   |ENT          |C|153     | 00038099 |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 7 IS:
 C(117:118,*,119)  C(115:122,*,123)  A(3,*,122)  A(6,*,123)  A(4,115,115)
 A(2,115,115)  A(1,*,117)

 CURRENT POOL FOR BLOCK# 9 IS:
 C(117:118,*,119)  C(114:120,*,121)  A(1,*,121)  A(3,*,120)  A(2,114,114)
AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 9 IS:
 A(1,*,124)  A(3,*,125)  A(2,*,126)  C(126:125,*,127)

 CONTROL AT 84, FROM BLK 4 TO BLK 5 (PASS 1)
C+84   0   |ENT          |C|133     | 00038085 |
B+85   0   |LOC          | |2       | 00020002 |
B+86   0   |INT          | |3       | 00050003 |
B+87   0   |STO          | |0       | 00280000 |
B+88   0   |DEL          | |0       | 002A0000 |
B+89   0   |LOC          | |10      | 0002000A |
B+90   0   |LOC          | |3       | 00020003 |
B+91   0   |LOD          | |0       | 00270000 |
B+92   0   |LOC          | |2       | 00020002 |
B+93   0   |LOD          | |0       | 00270000 |
B+94   0   |MUL          | |0       | 00140000 |
B+95   0   |STO          | |0       | 00280000 |
B+96   0   |DEL          | |0       | 002A0000 |
B+97   0   |ENT          | |1       | 00030001 |
 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 10 IS:
 C(117:118,*,119)  C(128:116,*,129)  A(3,*,128)  A(10,*,129)  A(2,116,116)
 A(1,*,117)

 CONTROL AT 113, FROM BLK 2 TO BLK 3 (PASS 1)
C+113  0   |ENT          |C|123     | 0003807B |
 CONTROL AT 68, FROM BLK 7 TO BLK 9 (PASS 1)
C+68   0   |ENT          |C|153     | 00038099 |
B+69   0   |LOC          | |5       | 00020005 |
B+70   0   |LOC          | |2       | 00020002 |
B+71   0   |LOD          | |0       | 00270000 |
B+72   0   |STO          | |0       | 00280000 |
B+73   0   |DEL          | |0       | 002A0000 |
B+74   0   |LOC          | |8       | 00020008 |
B+75   0   |LOC          | |5       | 00020005 |
B+76   0   |LOD          | |0       | 00270000 |
B+77   0   |LOC          | |3       | 00020003 |
B+78   0   |LOD          | |0       | 00270000 |
B+79   0   |MUL          | |0       | 00140000 |
B+80   0   |STO          | |0       | 00280000 |
B+81   0   |DEL          | |0       | 002A0000 |
B+82   0   |XIT          | |97      | 00040061 |
B+83   0   |BRS          | |0       | 002E0000 |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 9 IS:
 C(126:125,*,127)  A(8,*,127)  A(5,*,126)  A(1,*,124)  A(3,*,125)  A(2,*,126)

 CURRENT POOL FOR BLOCK# 10 IS:
 C(117:118,*,119)  C(128:116,*,129)  A(3,*,128)  A(10,*,129)  A(2,116,116)
 A(1,*,117)
AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 10 IS:
 A(3,*,130)  A(2,*,131)  A(1,*,132)  C(131:130,*,133)

 CONTROL AT 97, FROM BLK 9 TO BLK 10 (PASS 1)
C+97   0   |ENT          |C|158     | 0003809E |
B+98   0   |LOC          | |5       | 00020005 |
B+99   0   |LOC          | |3       | 00020003 |
```

```
B+100   0   |LOD          |   |0        |  00270000  |
B+101   0   |STO          |   |0        |  00280000  |
B+102   0   |DEL          |   |0        |  002A0000  |
B+103   0   |LOC          |   |9        |  00020009  |
B+104   0   |LOC          |   |2        |  00020002  |
B+105   0   |LOD          |   |0        |  00270000  |
B+106   0   |LOC          |   |3        |  00020003  |
B+107   0   |LOD          |   |0        |  00270000  |
B+108   0   |MUL          |   |0        |  00140000  |
B+109   0   |STO          |   |0        |  00280000  |
B+110   0   |DEL          |   |0        |  002A0000  |
B+111   0   |XIT          |   |8        |  00040008  |
B+112   0   |BRS          |   |0        |  002E0000  |
BEFORE THE MEET OPERATION:

  OUTPUT POOL FROM BLOCK# 10 IS:
  C(131:130,*,133)   A(9,*,133)   A(5,*,130)   A(3,*,130)   A(2,*,131)   A(1,*,132)

  CURRENT POOL FOR BLOCK# 2 IS:
  A(2,*,118)   A(1,*,117)

AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL


                                FINAL OPTIMIZATION RESULTS
##########################################################################################
------------------------------------------------------------------------------------------

OPTIMIZATION AT 22
          (117,118|13,119)     COMM SUBEXP ELIM

OPTIMIZATION AT 31
          (117,118|13,119)     COMM SUBEXP ELIM

OPTIMIZATION AT 79
          (126,125|79,127)     COMM SUBEXP ELIM

OPTIMIZATION AT 108
          (130,131|108,133)    COMM SUBEXP ELIM

  ****  FORMATTED INTERMEDIATE CODE DUMP  ****
  LOC OFF     OP CODE     CNS ADR     RAW CODE        OPTIMIZATION
      SET                     /ETC                         TYPE
*********************************************************************************
 0      0   |ENT          |   |114      |  00030072  |
 1      0   |TOGGLE       |   |257      |  000B0101  |
 2      0   |LOC          |   |1        |  00020001  |
 3      0   |INT          |   |0        |  00050000  |
 4      0   |STD          |   |0        |  00290000  |
 5      0   |LOC          |   |2        |  00020002  |
 6      0   |INT          |   |0        |  00050000  |
 7      0   |STD          |   |0        |  00290000  |
 8      0   |ENT          |   |118      |  00030076  |
 9      0   |LOC          |   |1        |  00020001  |
10      0   |LOD          |   |0        |  00270000  |
11      0   |LOC          |   |2        |  00020002  |
12      0   |LOD          |   |0        |  00270000  |
13      0   |GTR          |   |1024     |  00210400  |
14      0   |XIT          |   |113      |  00040071  |
15      0   |XIT          |   |17       |  00040011  |
16      0   |BSC          |   |0        |  002F0000  |
17      0   |ENT          |   |128      |  00030080  |
18      0   |LOC          |   |1        |  00020001  |
19      0   |LOD          |   |0        |  00270000  |
20      0   |LOC          |   |2        |  00020002  |
21      0   |LOD          |   |0        |  00270000  |
22      0   |GTR          |C  |1016     |  002183F8  |   COMM SUBEXP ELIM
23      0   |XIT          |   |84       |  00040054  |
24      0   |XIT          |   |26       |  0004001A  |
25      0   |BSC          |   |0        |  002F0000  |
26      0   |ENT          |   |138      |  0003008A  |
27      0   |LOC          |   |1        |  00020001  |
28      0   |LOD          |   |0        |  00270000  |
29      0   |LOC          |   |2        |  00020002  |
30      0   |LOD          |   |0        |  00270000  |
31      0   |GTR          |C  |1008     |  002183F0  |   COMM SUBEXP ELIM
32      0   |XIT          |   |50       |  00040032  |
33      0   |XIT          |   |35       |  00040023  |
34      0   |BSC          |   |0        |  002F0000  |
35      0   |ENT          |   |148      |  00030094  |
36      0   |LOC          |   |2        |  00020002  |
37      0   |INT          |   |1        |  00050001  |
```

38	0	STO		0	00280000	
39	0	DEL		0	002A0000	
40	0	LOC		1	00020001	
41	0	LOC		2	00020002	
42	0	LOD		0	00270000	
43	0	LOC		3	00020003	
44	0	LOD		0	00270000	
45	0	MUL		1000	001403E8	
46	0	STO		0	00280000	
47	0	DEL		0	002A0000	
48	0	XIT		68	00040044	
49	0	BRS		0	002E0000	
50	0	ENT		143	0003008F	
51	0	LOC		2	00020002	
52	0	INT		2	00050002	
53	0	STO		0	00280000	
54	0	DEL		0	002A0000	
55	0	LOC		4	00020004	
56	0	LOC		2	00020002	
57	0	LOD		0	00270000	
58	0	STO		0	00280000	
59	0	DEL		0	002A0000	
60	0	LOC		6	00020006	
61	0	LOC		4	00020004	
62	0	LOD		0	00270000	
63	0	LOC		3	00020003	
64	0	LOD		0	00270000	
65	0	MUL		992	001403E0	
66	0	STO		0	00280000	
67	0	DEL		0	002A0000	
68	0	ENT		153	00030099	
69	0	LOC		5	00020005	
70	0	LOC		2	00020002	
71	0	LOD		0	00270000	
72	0	STO		0	00280000	
73	0	DEL		0	002A0000	
74	0	LOC		8	00020008	
75	0	LOC		5	00020005	
76	0	LOD		0	00270000	
77	0	LOC		3	00020003	
78	0	LOD		0	00270000	
79	0	MUL	C	980	001483D4	COMM SUBEXP ELIM
80	0	STO		0	00280000	
81	0	DEL		0	002A0000	
82	0	XIT		97	00040061	
83	0	BRS		0	002E0000	
84	0	ENT		133	00030085	
85	0	LOC		2	00020002	
86	0	INT		3	00050003	
87	0	STO		0	00280000	
88	0	DEL		0	002A0000	
89	0	LOC		10	0002000A	
90	0	LOC		3	00020003	
91	0	LOD		0	00270000	
92	0	LOC		2	00020002	
93	0	LOD		0	00270000	
94	0	MUL		988	001403DC	
95	0	STO		0	00280000	
96	0	DEL		0	002A0000	
97	0	ENT		158	0003009E	
98	0	LOC		5	00020005	
99	0	LOC		3	00020003	
100	0	LOD		0	00270000	
101	0	STO		0	00280000	
102	0	DEL		0	002A0000	
103	0	LOC		9	00020009	
104	0	LOC		2	00020002	
105	0	LOD		0	00270000	
106	0	LOC		3	00020003	
107	0	LOD		0	00270000	
108	0	MUL	C	976	001483D0	COMM SUBEXP ELIM
109	0	STO		0	00280000	
110	0	DEL		0	002A0000	
111	0	XIT		8	00040008	
112	0	BRS		0	002E0000	
113	0	ENT		123	0003007B	

****

FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

FINAL POOL FOR BLOCK# 1 IS:
(NULL)

```
FINAL POCL FOR  BLOCK# 2 IS:
A(2,*,134)  A(1,*,135)


FINAL POCL FOR  BLOCK# 3 IS:
C(117:118,*,119)  A(2,*,118)  A(1,*,117)


FINAL POCL FOR  BLOCK# 4 IS:
C(117:118,*,119)  A(2,*,118)  A(1,*,117)


FINAL POOL FOR  BLOCK# 5 IS:
C(117:118,*,119)  A(2,*,118)  A(1,*,117)


FINAL POCL FOR  BLOCK# 6 IS:
C(117:118,*,119)  A(2,*,118)  A(1,*,117)


FINAL POCL FOR  BLOCK# 7 IS:
C(117:118,*,119)  A(2,*,118)  A(1,*,117)


FINAL POCL FOR  BLOCK# 8 IS:
C(117:118,*,119)  A(2,*,118)  A(1,*,117)


FINAL PCCL FOR  BLOCK# 9 IS:
A(1,*,124)  A(3,*,125)  A(2,*,126)  C(126:125,*,127)


FINAL POCL FOR  BLOCK# 10 IS:
A(3,*,130)  A(2,*,131)  A(1,*,132)  C(131:130,*,133)


    BASIC BLOCK #1
          BEGINNING AT 0, ENDING AT 8
          BLOCK TRAVERSED 1 TIMES
          0 REFERENCES:
    BASIC BLOCK #2
          BEGINNING AT 8, ENDING AT 16
          BLOCK TRAVERSED 1 TIMES
          1 REFERENCES:
              1
    BASIC BLOCK #3
          BEGINNING AT 113, ENDING AT 113
          BLOCK TRAVERSED 1 TIMES
          1 REFERENCES:
              2
    BASIC BLOCK #4
          BEGINNING AT 17, ENDING AT 25
          BLOCK TRAVERSED 1 TIMES
          1 REFERENCES:
              2
    BASIC BLOCK #5
          BEGINNING AT 84, ENDING AT 97
          BLOCK TRAVERSED 1 TIMES
          1 REFERENCES:
              4
    BASIC BLOCK #6
          BEGINNING AT 26, ENDING AT 34
          BLOCK TRAVERSED 1 TIMES
          1 REFERENCES:
              4
    BASIC BLOCK #7
          BEGINNING AT 50, ENDING AT 68
          BLOCK TRAVERSED 1 TIMES
          1 REFERENCES:
              6
    BASIC BLOCK #8
          BEGINNING AT 35, ENDING AT 49
          BLOCK TRAVERSED 1 TIMES
          1 REFERENCES:
              6
    BASIC BLOCK #9
          BEGINNING AT 68, ENDING AT 83
```

```
        BLOCK TRAVERSED 1 TIMES
        1 REFERENCES:
           7
  BASIC BLOCK #10
        BEGINNING AT 97, ENDING AT 112
        BLOCK TRAVERSED 1 TIMES
        1 REFERENCES:
           9

BLOCK SUMMARY DATA:

BLOCK NUMBER                          NUMBER OF PASSES

     1                                      1
     2                                      1
     3                                      1
     4                                      1
     5                                      1
     6                                      1
     7                                      1
     8                                      1
     9                                      1
    10                                      1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 10
USING THE STEEPEST DESCENT (MINIMUM CURRENT POOL)  BLOCK SELECTION ALGORITHM.

TIME FOR THE OPTIMIZATION WAS 0.267 SECONDS.

                    ***CODE SYNTHESIS FILTER IS COMPLETE***
```

```
***CODE SYNTHESIS FILTER***

BLOCK SELECTION METHOD WILL BE FIRST-IN-FIRST-OUT

***C O M P L E T E   T A B L E   D U M P***
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

***T A B L E   D U M P*** AT LOCATION = 0

CONSTANTS <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
CONSYM   |10  |20  |30
CONTYPE  |INT |INT |INT
CONVAL   |114 |115 |116
CONINT   |10  |20  |30

ADDR TABL <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

VALUE STACK (TOP AT 127)

CONT STK  <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
BLOCK#   |0
ENTRY    |0

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)
```

```
 ****  FORMATTED INTERMEDIATE CODE DUMP   ****
 LOC OFF     OP CODE     CNS ADR      RAW CODE      OPTIMIZATION
     SET                     /ETC                        TYPE
************************************************************************************
 0      0    |ENT        |C|114     | 00038072 |
 1      0    |TOGGLE     | |257     | 000B0101 |
 2      0    |LOC        | |1       | 00020001 |
 3      0    |INT        | |0       | 00050000 |
 4      0    |STD        | |0       | 00290000 |
 5      0    |LOC        | |2       | 00020002 |
 6      0    |INT        | |0       | 00050000 |
 7      0    |STD        | |0       | 00290000 |
 8      0    |ENT        | |1       | 00030001 |
 9      0    |LOC        | |1       | 00020001 |
 10     0    |LOD        | |0       | 00270000 |
 11     0    |LOC        | |2       | 00020002 |
 12     0    |LOD        | |0       | 00270000 |
 13     0    |GTR        | |0       | 00210000 |
 14     0    |XIT        | |113     | 00040071 |
 15     0    |XIT        | |17      | 00040011 |
 16     0    |BSC        | |0       | 002F0000 |
 17     0    |ENT        | |1       | 00030001 |
 18     0    |LOC        | |1       | 00020001 |
 19     0    |LOD        | |0       | 00270000 |
 20     0    |LOC        | |2       | 00020002 |
 21     0    |LOD        | |0       | 00270000 |
 22     0    |GTR        | |0       | 00210000 |
 23     0    |XIT        | |84      | 00040054 |
 24     0    |XIT        | |26      | 0004001A |
 25     0    |BSC        | |0       | 002F0000 |
 26     0    |ENT        | |1       | 00030001 |
 27     0    |LOC        | |1       | 00020001 |
 28     0    |LOD        | |0       | 00270000 |
 29     0    |LOC        | |2       | 00020002 |
 30     0    |LOD        | |0       | 00270000 |
 31     0    |GTR        | |0       | 00210000 |
 32     0    |XIT        | |50      | 00040032 |
 33     0    |XIT        | |35      | 00040023 |
 34     0    |BSC        | |0       | 002F0000 |
 35     0    |ENT        | |1       | 00030001 |
 36     0    |LOC        | |2       | 00020002 |
 37     0    |INT        | |1       | 00050001 |
 38     0    |STO        | |0       | 00280000 |
 39     0    |DEL        | |0       | 002A0000 |
 40     0    |LOC        | |1       | 00020001 |
 41     0    |LOC        | |2       | 00020002 |
```

42	0	LOD	0	00270000
43	0	LOC	3	00020003
44	0	LOD	0	00270000
45	0	MUL	0	00140000
46	0	STO	0	00280000
47	0	DEL	0	002A0000
48	0	XIT	68	00040044
49	0	BRS	0	002E0000
50	0	ENT	1	00030001
51	0	LOC	2	00020002
52	0	INT	2	00050002
53	0	STO	0	00280000
54	0	DEL	0	002A0000
55	0	LOC	4	00020004
56	0	LOC	2	00020002
57	0	LOD	0	00270000
58	0	STO	0	00280000
59	0	DEL	0	002A0000
60	0	LOC	6	00020006
61	0	LOC	4	00020004
62	0	LOD	0	00270000
63	0	LOC	3	00020003
64	0	LOD	0	00270000
65	0	MUL	0	00140000
66	0	STO	0	00280000
67	0	DEL	0	002A0000
68	0	ENT	1	00030001
69	0	LOC	5	00020005
70	0	LOC	2	00020002
71	0	LOD	0	00270000
72	0	STO	0	00280000
73	0	DEL	0	002A0000
74	0	LOC	8	00020008
75	0	LOC	5	00020005
76	0	LOD	0	00270000
77	0	LOC	3	00020003
78	0	LOD	0	00270000
79	0	MUL	0	00140000
80	0	STO	0	00280000
81	0	DEL	0	002A0000
82	0	XIT	97	00040061
83	0	BRS	0	002E0000
84	0	ENT	1	00030001
85	0	LOC	2	00020002
86	0	INT	3	00050003
87	0	STO	0	00280000
88	0	DEL	0	002A0000
89	0	LOC	10	0002000A
90	0	LOC	3	00020003
91	0	LOD	0	00270000
92	0	LOC	2	00020002
93	0	LOD	0	00270000
94	0	MUL	0	00140000
95	0	STO	0	00280000
96	0	DEL	0	002A0000
97	0	ENT	1	00030001
98	0	LOC	5	00020005
99	0	LOC	3	00020003
100	0	LOD	0	00270000
101	0	STO	0	00280000
102	0	DEL	0	002A0000
103	0	LOC	9	00020009
104	0	LOC	2	00020002
105	0	LOD	0	00270000
106	0	LOC	3	00020003
107	0	LOD	0	00270000
108	0	MUL	0	00140000
109	0	STO	0	00280000
110	0	DEL	0	002A0000
111	0	XIT	8	00040008
112	0	BRS	0	002E0000
113	0	ENT	1	00030001

****

BASIC BLOCK #1
BEGINNING AT 0, ENDING AT 0
BLOCK TRAVERSED 0 TIMES
0 REFERENCES:

E N D O F C O M P L E T E T A B L E D U M P

B+2	0	LOC	1	00020001
B+3	0	INT	0	00050000
B+4	0	STD	0	00290000
B+5	0	LOC	2	00020002
B+6	0	INT	0	00050000
B+7	0	STD	0	00290000

```
B+8     0    |ENT            | 11      | 00030001  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 2 IS:
 A(2,*,118)  A(1,*,117)

 CONTROL AT 8, FROM BLK 1 TO BLK 2 (PASS 1)
C+8     0    |ENT           |C|118     | 00038076  |
B+9     0    |LOC           | |1       | 00020001  |
B+10    0    |LOD           | |0       | 00270000  |
B+11    0    |LOC           | |2       | 00020002  |
B+12    0    |LOD           | |0       | 00270000  |
B+13    0    |GTR           | |0       | 00210000  |
B+14    0    |XIT           | |113     | 00040071  |
B+15    0    |XIT           | |17      | 00040011  |
B+16    0    |BSC           | |0       | 002F0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 3 IS:
 C(117:118,*,119)  A(2,*,118)  A(1,*,117)


 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 4 IS:
 C(117:118,*,119)  A(2,*,118)  A(1,*,117)

 CONTROL AT 113, FROM BLK 2 TO BLK 3 (PASS 1)
C+113   0    |ENT           |C|123     | 0003807B  |
 CONTROL AT 17, FROM BLK 2 TO BLK 4 (PASS 1)
C+17    0    |ENT           |C|128     | 00038080  |
B+18    0    |LOC           | |1       | 00020001  |
B+19    0    |LOD           | |0       | 00270000  |
B+20    0    |LOC           | |2       | 00020002  |
B+21    0    |LOD           | |0       | 00270000  |
B+22    0    |GTR           | |0       | 00210000  |
B+23    0    |XIT           | |84      | 00040054  |
B+24    0    |XIT           | |26      | 0004001A  |
B+25    0    |BSC           | |0       | 002F0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 5 IS:
 C(117:118,*,119)  A(2,*,118)  A(1,*,117)


 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 6 IS:
 C(117:118,*,119)  A(2,*,118)  A(1,*,117)

 CONTROL AT 84, FROM BLK 4 TO BLK 5 (PASS 1)
C+84    0    |ENT           |C|133     | 00038085  |
B+85    0    |LOC           | |2       | 00020002  |
B+86    0    |INT           | |3       | 00050003  |
B+87    0    |STO           | |0       | 00280000  |
B+88    0    |DEL           | |0       | 002A0000  |
B+89    0    |LOC           | |10      | 0002000A  |
B+90    0    |LOC           | |3       | 00020003  |
B+91    0    |LOD           | |0       | 00270000  |
B+92    0    |LOC           | |2       | 00020002  |
B+93    0    |LOD           | |0       | 00270000  |
B+94    0    |MUL           | |0       | 00140000  |
B+95    0    |STO           | |0       | 00280000  |
B+96    0    |DEL           | |0       | 002A0000  |
B+97    0    |ENT           | |1       | 00030001  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 7 IS:
 C(117:118,*,119)  C(120:116,*,121)  A(3,*,120)  A(10,*,121)  A(2,116,116)
 A(1,*,117)

 CONTROL AT 26, FROM BLK 4 TO BLK 6 (PASS 1)
C+26    0    |ENT           |C|138     | 0003808A  |
B+27    0    |LOC           | |1       | 00020001  |
B+28    0    |LOD           | |0       | 00270000  |
B+29    0    |LOC           | |2       | 00020002  |
B+30    0    |LOD           | |0       | 00270000  |
B+31    0    |GTR           | |0       | 00210000  |
B+32    0    |XIT           | |50      | 00040032  |
B+33    0    |XIT           | |35      | 00040023  |
B+34    0    |BSC           | |0       | 002F0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 8 IS:
 C(117:118,*,119)  A(2,*,118)  A(1,*,117)


 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 9 IS:
 C(117:118,*,119)  A(2,*,118)  A(1,*,117)

 CONTROL AT 97, FROM BLK 5 TO BLK 7 (PASS 1)
C+97    0    |ENT           |C|143     | 0003808F  |
B+98    0    |LOC           | |5       | 00020005  |
B+99    0    |LOC           | |3       | 00020003  |
B+100   0    |LOD           | |0       | 00270000  |
B+101   0    |STO           | |0       | 00280000  |
B+102   0    |DEL           | |0       | 002A0000  |
```

```
B+103  0   |LOC            | |9        | 00020009 |
B+104  0   |LOC            | |2        | 00020002 |
B+105  0   |LOD            | |0        | 00270000 |
B+106  0   |LOC            | |3        | 00020003 |
B+107  0   |LOD            | |0        | 00270000 |
B+108  0   |MUL            | |0        | 00140000 |
B+109  0   |STO            | |0        | 00280000 |
B+110  0   |DEL            | |0        | 002A0000 |
B+111  0   |XIT            | |8        | 00040008 |
B+112  0   |BRS            | |0        | 002E0000 |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 7 IS:
 C(120:116,*,121)  C(117:118,*,119)  A(9,*,121)  A(5,*,120)  A(3,*,120)
 A(10,*,121)  A(2,116,116)  A(1,*,117)


 CURRENT POOL FOR BLOCK# 2 IS:
 A(2,*,118)  A(1,*,117)

AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

 CONTROL AT 50, FROM BLK 6 TO BLK 8 (PASS 1)
C+50   0   |ENT            |C|148      | 00038094 |
B+51   0   |LOC            | |2        | 00020002 |
B+52   0   |INT            | |2        | 00050002 |
B+53   0   |STO            | |0        | 00280000 |
B+54   0   |DEL            | |0        | 002A0000 |
B+55   0   |LOC            | |4        | 00020004 |
B+56   0   |LOC            | |2        | 00020002 |
B+57   0   |LOD            | |0        | 00270000 |
B+58   0   |STO            | |0        | 00280000 |
B+59   0   |DEL            | |0        | 002A0000 |
B+60   0   |LOC            | |6        | 00020006 |
B+61   0   |LOC            | |4        | 00020004 |
B+62   0   |LOD            | |0        | 00270000 |
B+63   0   |LOC            | |3        | 00020003 |
B+64   0   |LOD            | |0        | 00270000 |
B+65   0   |MUL            | |0        | 00140000 |
B+66   0   |STO            | |0        | 00280000 |
B+67   0   |DEL            | |0        | 002A0000 |
B+68   0   |ENT            | |1        | 00030001 |
 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 10 IS:
 C(117:118,*,119)  C(115:123,*,124)  A(3,*,123)  A(6,*,124)  A(4,115,115)
 A(2,115,115)  A(1,*,117)

 CONTROL AT 35, FROM BLK 6 TO BLK 9 (PASS 1)
C+35   0   |ENT            |C|153      | 00038099 |
B+36   0   |LOC            | |2        | 00020002 |
B+37   0   |INT            | |1        | 00050001 |
B+38   0   |STO            | |0        | 00280000 |
B+39   0   |DEL            | |0        | 002A0000 |
B+40   0   |LOC            | |1        | 00020001 |
B+41   0   |LOC            | |2        | 00020002 |
B+42   0   |LOD            | |0        | 00270000 |
B+43   0   |LOC            | |3        | 00020003 |
B+44   0   |LOD            | |0        | 00270000 |
B+45   0   |MUL            | |0        | 00140000 |
B+46   0   |STO            | |0        | 00280000 |
B+47   0   |DEL            | |0        | 002A0000 |
B+48   0   |XIT            | |68       | 00040044 |
B+49   0   |BRS            | |0        | 002E0000 |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 9 IS:
 C(117:118,*,119)  C(114:125,*,126)  A(1,*,126)  A(3,*,125)  A(2,114,114)


 CURRENT POOL FOR BLOCK# 10 IS:
 C(117:118,*,119)  C(115:123,*,124)  A(3,*,123)  A(6,*,124)  A(4,115,115)
 A(2,115,115)  A(1,*,117)
AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 10 IS:
 A(3,*,127)  A(2,*,128)  A(1,*,129)


 CONTROL AT 68, FROM BLK 9 TO BLK 10 (PASS 1)
C+68   0   |ENT            |C|158      | 0003809E |
B+69   0   |LOC            | |5        | 00020005 |
B+70   0   |LOC            | |2        | 00020002 |
B+71   0   |LOD            | |0        | 00270000 |
B+72   0   |STO            | |0        | 00280000 |
```

```
B+73    0   |DEL           | |0        | 002A0000  |
B+74    0   |LOC           | |8        | 00020008  |
B+75    0   |LOC           | |5        | 00020005  |
B+76    0   |LOD           | |0        | 00270000  |
B+77    0   |LOC           | |3        | 00020003  |
B+78    0   |LOD           | |0        | 00270000  |
B+79    0   |MUL           | |0        | 00140000  |
B+80    0   |STO           | |0        | 00280000  |
B+81    0   |DEL           | |0        | 002A0000  |
B+82    0   |XIT           | |97       | 00040061  |
B+83    0   |BRS           | |0        | 002E0000  |
BEFORE THE MEET OPERATION:

  OUTPUT POOL FROM BLOCK# 10 IS:
  C(128:127,*,130)  A(8,*,130)  A(5,*,128)  A(3,*,127)  A(2,*,128)  A(1,*,129)

  CURRENT POOL FOR BLOCK# 7 IS:
  C(117:118,*,119)  C(120:116,*,121)  A(3,*,120)  A(10,*,121)  A(2,116,116)
  A(1,*,117)
AFTER PERFORMING THE MEET OPERATION:

  NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 7 IS:
  A(3,*,131)  A(2,*,132)  A(1,*,133)  C(132:131,*,134)


  CONTROL AT 97, FROM BLK 10 TO BLK 7 (PASS 2)
C+97    0   |ENT           |C|143      | 0003808F  |
B+98    0   |LOC           | |5        | 00020005  |
B+99    0   |LOC           | |3        | 00020003  |
B+100   0   |LOD           | |0        | 00270000  |
B+101   0   |STO           | |0        | 00280000  |
B+102   0   |DEL           | |0        | 002A0000  |
B+103   0   |LOC           | |9        | 00020009  |
B+104   0   |LOC           | |2        | 00020002  |
B+105   0   |LOD           | |0        | 00270000  |
B+106   0   |LOC           | |3        | 00020003  |
B+107   0   |LOD           | |0        | 00270000  |
B+108   0   |MUL           |C|992      | 001483E0  |  COMM SUBEXP ELIM
B+109   0   |STO           | |0        | 00280000  |
B+110   0   |DEL           | |0        | 002A0000  |
B+111   0   |XIT           | |8        | 00040008  |
B+112   0   |BRS           | |0        | 002E0000  |
BEFORE THE MEET OPERATION:

  OUTPUT POOL FROM BLOCK# 7 IS:
  C(132:131,*,134)  A(9,*,134)  A(5,*,131)  A(3,*,131)  A(2,*,132)  A(1,*,133)

  CURRENT POOL FOR BLOCK# 2 IS:
  A(2,*,122)  A(1,*,117)

AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL


                               FINAL OPTIMIZATION RESULTS
##################################################################################
----------------------------------------------------------------------------------

OPTIMIZATION AT 22
          (117,118|13,119)     COMM SUBEXP ELIM

OPTIMIZATION AT 31
          (117,118|13,119)     COMM SUBEXP ELIM

OPTIMIZATION AT 108
          (132,131|94,134)     COMM SUBEXP ELIM


  ****  FORMATTED INTERMEDIATE CODE DUMP   ****
  LOC OFF     OP CODE     CNS ADR     RAW CODE       OPTIMIZATION
      SET                  /ETC                          TYPE
**********************************************************************
0       0   |ENT           | |114      | 00030072  |
1       0   |TOGGLE        | |257      | 00080101  |
2       0   |LOC           | |1        | 00020001  |
3       0   |INT           | |0        | 00050000  |
4       0   |STD           | |0        | 00290000  |
5       0   |LOC           | |2        | 00020002  |
6       0   |INT           | |0        | 00050000  |
7       0   |STD           | |0        | 00290000  |
8       0   |ENT           | |118      | 00030076  |
```

9	0	LOC		1	00020001	
10	0	LOD		0	00270000	
11	0	LOC		2	00020002	
12	0	LOD		0	00270000	
13	0	GTR		1024	00210400	
14	0	XIT		113	00040071	
15	0	XIT		17	00040011	
16	0	BSC		0	002F0000	
17	0	ENT		128	00030080	
18	0	LOC		1	00020001	
19	0	LOD		0	00270000	
20	0	LOC		2	00020002	
21	0	LOD		0	00270000	
22	0	GTR	C	1016	002183F8	COMM SUBEXP ELIM
23	0	XIT		84	00040054	
24	0	XIT		26	0004001A	
25	0	BSC		0	002F0000	
26	0	ENT		138	0003008A	
27	0	LOC		1	00020001	
28	0	LOD		0	00270000	
29	0	LOC		2	00020002	
30	0	LOD		0	00270000	
31	0	GTR	C	1000	002183E8	COMM SUBEXP ELIM
32	0	XIT		50	00040032	
33	0	XIT		35	00040023	
34	0	BSC		0	002F0000	
35	0	ENT		153	00030099	
36	0	LOC		2	00020002	
37	0	INT		1	00050001	
38	0	STO		0	00280000	
39	0	DEL		0	002A0000	
40	0	LOC		1	00020001	
41	0	LOC		2	00020002	
42	0	LOD		0	00270000	
43	0	LOC		3	00020003	
44	0	LOD		0	00270000	
45	0	MUL		980	001403D4	
46	0	STO		0	00280000	
47	0	DEL		0	002A0000	
48	0	XIT		68	00040044	
49	0	BRS		0	002E0000	
50	0	ENT		148	00030094	
51	0	LOC		2	00020002	
52	0	INT		2	00050002	
53	0	STO		0	00280000	
54	0	DEL		0	002A0000	
55	0	LOC		4	00020004	
56	0	LOC		2	00020002	
57	0	LOD		0	00270000	
58	0	STO		0	00280000	
59	0	DEL		0	002A0000	
60	0	LOC		6	00020006	
61	0	LOC		4	00020004	
62	0	LOD		0	00270000	
63	0	LOC		3	00020003	
64	0	LOD		0	00270000	
65	0	MUL		988	0014030C	
66	0	STO		0	00280000	
67	0	DEL		0	002A0000	
68	0	ENT		158	0003009E	
69	0	LOC		5	00020005	
70	0	LOC		2	00020002	
71	0	LOD		0	00270000	
72	0	STO		0	00280000	
73	0	DEL		0	002A0000	
74	0	LOC		8	00020008	
75	0	LOC		5	00020005	
76	0	LOD		0	00270000	
77	0	LOC		3	00020003	
78	0	LOD		0	00270000	
79	0	MUL		976	001403D0	
80	0	STO		0	00280000	
81	0	DEL		0	002A0000	
82	0	XIT		97	00040061	
83	0	BRS		0	002E0000	
84	0	ENT		133	00030085	
85	0	LOC		2	00020002	
86	0	INT		3	00050003	
87	0	STO		0	00280000	
88	0	DEL		0	002A0000	
89	0	LOC		10	0002000A	
90	0	LOC		3	00020003	
91	0	LOD		0	00270000	
92	0	LOC		2	00020002	
93	0	LOD		0	00270000	
94	0	MUL		1008	001403F0	
95	0	STO		0	00280000	
96	0	DEL		0	002A0000	

```
97    0   ENT      143    0003008F
98    0   LOC      5      00020005
99    0   LOC      3      00020003
100   0   LOD      0      00270000
101   0   STO      0      00280000
102   0   DEL      0      002A0000
103   0   LOC      9      00020009
104   0   LOC      2      00020002
105   0   LOD      0      00270000
106   0   LOC      3      00020003
107   0   LOD      0      00270000
108   0   MUL    C 992    001483E0    COMM SUBEXP ELIM
109   0   STO      0      00280000
110   0   DEL      0      002A0000
111   0   XIT      8      00040008
112   0   BRS      0      002E0000
113   0   ENT      123    0003007B
*************************************************************************************
```

```
        FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

 FINAL POOL FOR  BLOCK# 1 IS:
(NULL)


 FINAL POOL FOR  BLOCK# 2 IS:
 A(2,*,135)  A(1,*,136)


 FINAL POOL FOR  BLOCK# 3 IS:
 C(117:118,*,119)  A(2,*,118)   A(1,*,117)


 FINAL POOL FOR  BLOCK# 4 IS:
 C(117:118,*,119)  A(2,*,118)   A(1,*,117)


 FINAL POOL FOR  BLOCK# 5 IS:
 C(117:118,*,119)  A(2,*,118)   A(1,*,117)


 FINAL POOL FOR  BLOCK# 6 IS:
 C(117:118,*,119)  A(2,*,118)   A(1,*,117)


 FINAL POOL FOR  BLOCK# 7 IS:
 A(3,*,131)  A(2,*,132)  A(1,*,133)  C(132:131,*,134)


 FINAL POOL FOR  BLOCK# 8 IS:
 C(117:118,*,119)  A(2,*,118)   A(1,*,117)


 FINAL POOL FOR  BLOCK# 9 IS:
 C(117:118,*,119)  A(2,*,118)   A(1,*,117)


 FINAL POOL FOR  BLOCK# 10 IS:
 A(3,*,127)  A(2,*,128)  A(1,*,129)


      BASIC BLOCK #1
            BEGINNING AT 0, ENDING AT 8
            BLOCK TRAVERSED 1 TIMES
            0 REFERENCES:
      BASIC BLOCK #2
            BEGINNING AT 8, ENDING AT 16
            BLOCK TRAVERSED 1 TIMES
            1 REFERENCES:
               1
      BASIC BLOCK #3
            BEGINNING AT 113, ENDING AT 113
            BLOCK TRAVERSED 1 TIMES
```

```
        1 REFERENCES:
             2
   BASIC BLOCK #4
         BEGINNING AT 17, ENDING AT 25
         BLOCK TRAVERSED 1 TIMES
         1 REFERENCES:
             2
   BASIC BLOCK #5
         BEGINNING AT 84, ENDING AT 97
         BLOCK TRAVERSED 1 TIMES
         1 REFERENCES:
             4
   BASIC BLOCK #6
         BEGINNING AT 26, ENDING AT 34
         BLOCK TRAVERSED 1 TIMES
         1 REFERENCES:
             4
   BASIC BLOCK #7
         BEGINNING AT 97, ENDING AT 112
         BLOCK TRAVERSED 2 TIMES
         1 REFERENCES:
             5
   BASIC BLOCK #8
         BEGINNING AT 50, ENDING AT 68
         BLOCK TRAVERSED 1 TIMES
         1 REFERENCES:
             6
   BASIC BLOCK #9
         BEGINNING AT 35, ENDING AT 49
         BLOCK TRAVERSED 1 TIMES
         1 REFERENCES:
             6
   BASIC BLOCK #10
         BEGINNING AT 68, ENDING AT 83
         BLOCK TRAVERSED 1 TIMES
         1 REFERENCES:
             9

BLOCK SUMMARY DATA:

BLOCK NUMBER                     NUMBER OF PASSES

     1                                  1
     2                                  1
     3                                  1
     4                                  1
     5                                  1
     6                                  1
     7                                  2
     8                                  1
     9                                  1
     10                                 1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 11
USING THE FIRST-IN-FIRST-OUT  BLOCK SELECTION ALGORITHM.

TIME FOR THE OPTIMIZATION WAS 0.216 SECONDS.

                    ***CODE SYNTHESIS FILTER IS COMPLETE***
```

```
                    ***CODE SYNTHESIS FILTER***

BLOCK SELECTION METHOD WILL BE MAXIMUM CURRENT POOL

***C O M P L E T E   T A B L E   D U M P***
DUMP RECUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

***T A B L E   D U M P*** AT LOCATION = 0

CONSTANTS <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
CONSYM   |10  |20  |30
CONTYPE  |INT |INT |INT
CONVAL   |114 |115 |116
CONINT   |10  |20  |30

ADDR TABL <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

VALUE STACK (TOP AT 127)


CONT STK  <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
BLOCK#   |0
ENTRY    |0

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)


 ****  FORMATTED INTERMEDIATE CODE DUMP  ****
 LOC OFF     OP CODE    CNS ADR    RAW CODE      OPTIMIZATION
     SET                  /ETC                       TYPE
*************************************************************************************
 0     0    |ENT       |C|114    | 00038072 |
 1     0    |TOGGLE    | |257    | 000B0101 |
 2     0    |LOC       | |1      | 00020001 |
 3     0    |INT       | |0      | 00050000 |
 4     0    |STD       | |0      | 00290000 |
 5     0    |LOC       | |2      | 00020002 |
 6     0    |INT       | |0      | 00050000 |
 7     0    |STD       | |0      | 00290000 |
 8     0    |ENT       | |1      | 00030001 |
 9     0    |LOC       | |1      | 00020001 |
 10    0    |LOD       | |0      | 00270000 |
 11    0    |LOC       | |2      | 00020002 |
 12    0    |LOD       | |0      | 00270000 |
 13    0    |GTR       | |0      | 00210000 |
 14    0    |XIT       | |113    | 00040071 |
 15    0    |XIT       | |17     | 00040011 |
 16    0    |BSC       | |0      | 002F0000 |
 17    0    |ENT       | |1      | 00030001 |
 18    0    |LOC       | |1      | 00020001 |
 19    0    |LOD       | |0      | 00270000 |
 20    0    |LOC       | |2      | 00020002 |
 21    0    |LOD       | |0      | 00270000 |
 22    0    |GTR       | |0      | 00210000 |
 23    0    |XIT       | |84     | 00040054 |
 24    0    |XIT       | |26     | 0004001A |
 25    0    |BSC       | |0      | 002F0000 |
 26    0    |ENT       | |1      | 00030001 |
 27    0    |LOC       | |1      | 00020001 |
 28    0    |LOD       | |0      | 00270000 |
 29    0    |LOC       | |2      | 00020002 |
 30    0    |LOD       | |0      | 00270000 |
 31    0    |GTR       | |0      | 00210000 |
 32    0    |XIT       | |50     | 00040032 |
 33    0    |XIT       | |35     | 00040023 |
 34    0    |BSC       | |0      | 002F0000 |
 35    0    |ENT       | |1      | 00030001 |
 36    0    |LOC       | |2      | 00020002 |
 37    0    |INT       | |1      | 00050001 |
 38    0    |STO       | |0      | 00280000 |
 39    0    |DEL       | |0      | 002A0000 |
 40    0    |LOC       | |1      | 00020001 |
 41    0    |LOC       | |2      | 00020002 |
```

42	0	LOD	0	00270000
43	0	LOC	3	00020003
44	0	LOD	0	00270000
45	0	MUL	0	00140000
46	0	STO	0	00280000
47	0	DEL	0	002A0000
48	0	XIT	68	00040044
49	0	BRS	0	002E0000
50	0	ENT	1	00030001
51	0	LOC	2	00020002
52	0	INT	2	00050002
53	0	STO	0	00280000
54	0	DEL	0	002A0000
55	0	LOC	4	00020004
56	0	LOC	2	00020002
57	0	LOD	0	00270000
58	0	STO	0	00280000
59	0	DEL	0	002A0000
60	0	LOC	6	00020006
61	0	LOC	4	00020004
62	0	LOD	0	00270000
63	0	LOC	3	00020003
64	0	LOD	0	00270000
65	0	MUL	0	00140000
66	0	STO	0	00280000
67	0	DEL	0	002A0000
68	0	ENT	1	00030001
69	0	LOC	5	00020005
70	0	LOC	2	00020002
71	0	LOD	0	00270000
72	0	STO	0	00280000
73	0	DEL	0	002A0000
74	0	LOC	8	00020008
75	0	LOC	5	00020005
76	0	LOD	0	00270000
77	0	LOC	3	00020003
78	0	LOD	0	00270000
79	0	MUL	0	00140000
80	0	STO	0	00280000
81	0	DEL	0	002A0000
82	0	XIT	97	00040061
83	0	BRS	0	002E0000
84	0	ENT	1	00030001
85	0	LOC	2	00020002
86	0	INT	3	00050003
87	0	STO	0	00280000
88	0	DEL	0	002A0000
89	0	LOC	10	0002000A
90	0	LOC	3	00020003
91	0	LOD	0	00270000
92	0	LOC	2	00020002
93	0	LOD	0	00270000
94	0	MUL	0	00140000
95	0	STO	0	00280000
96	0	DEL	0	002A0000
97	0	ENT	1	00030001
98	0	LOC	5	00020005
99	0	LOC	3	00020003
100	0	LOD	0	00270000
101	0	STO	0	00280000
102	0	DEL	0	002A0000
103	0	LOC	9	00020009
104	0	LOC	2	00020002
105	0	LOD	0	00270000
106	0	LOC	3	00020003
107	0	LOD	0	00270000
108	0	MUL	0	00140000
109	0	STO	0	00280000
110	0	DEL	0	002A0000
111	0	XIT	8	00040008
112	0	BRS	0	002E0000
113	0	ENT	1	00030001

```
************************************************************************************************

      BASIC BLOCK #1
            BEGINNING AT 0, ENDING AT 0
            BLOCK TRAVERSED 0 TIMES
            0 REFERENCES:

***E N D    O F    C O M P L E T E    T A B L E    D U M P***
```

B+2	0	LOC	1	00020001
B+3	0	INT	0	00050000
B+4	0	STD	0	00290000
B+5	0	LOC	2	00020002
B+6	0	INT	0	00050000
B+7	0	STD	0	00290000

```
B+8      0    |ENT              | 11      | 00030001   |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 2 IS:
 A(2,*,118)   A(1,*,117)

 CONTROL AT 8, FROM BLK 1 TO BLK 2 (PASS 1)
C+8      0    |ENT             |C|118      | 00038076   |
B+9      0    |LOC             | |1        | 00020001   |
B+10     0    |LOD             | |0        | 00270000   |
B+11     0    |LOC             | |2        | 00020002   |
B+12     0    |LOD             | |0        | 00270000   |
B+13     0    |GTR             | |0        | 00210000   |
B+14     0    |XIT             | |113      | 00040071   |
B+15     0    |XIT             | |17       | 00040011   |
B+16     0    |BSC             | |0        | 002F0000   |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 3 IS:
 C(117:118,*,119)   A(2,*,118)   A(1,*,117)

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 4 IS:
 C(117:118,*,119)   A(2,*,118)   A(1,*,117)

 CONTROL AT 17, FROM BLK 2 TO BLK 4 (PASS 1)
C+17     0    |ENT             |C|128      | 00038080   |
B+18     0    |LOC             | |1        | 00020001   |
B+19     0    |LOD             | |0        | 00270000   |
B+20     0    |LOC             | |2        | 00020002   |
B+21     0    |LOD             | |0        | 00270000   |
B+22     0    |GTR             | |0        | 00210000   |
B+23     0    |XIT             | |84       | 00040054   |
B+24     0    |XIT             | |26       | 0004001A   |
B+25     0    |BSC             | |0        | 002F0000   |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 5 IS:
 C(117:118,*,119)   A(2,*,118)   A(1,*,117)

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 6 IS:
 C(117:118,*,119)   A(2,*,118)   A(1,*,117)

 CONTROL AT 26, FROM BLK 4 TO BLK 6 (PASS 1)
C+26     0    |ENT             |C|138      | 0003808A   |
B+27     0    |LOC             | |1        | 00020001   |
B+28     0    |LOD             | |0        | 00270000   |
B+29     0    |LOC             | |2        | 00020002   |
B+30     0    |LOD             | |0        | 00270000   |
B+31     0    |GTR             | |0        | 00210000   |
B+32     0    |XIT             | |50       | 00040032   |
B+33     0    |XIT             | |35       | 00040023   |
B+34     0    |BSC             | |0        | 002F0000   |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 7 IS:
 C(117:118,*,119)   A(2,*,118)   A(1,*,117)

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 8 IS:
 C(117:118,*,119)   A(2,*,118)   A(1,*,117)

 CONTROL AT 35, FROM BLK 6 TO BLK 8 (PASS 1)
C+35     0    |ENT             |C|148      | 00038094   |
B+36     0    |LOC             | |2        | 00020002   |
B+37     0    |INT             | |1        | 00050001   |
B+38     0    |STO             | |0        | 00280000   |
B+39     0    |DEL             | |0        | 002A0000   |
B+40     0    |LOC             | |1        | 00020001   |
B+41     0    |LOC             | |2        | 00020002   |
B+42     0    |LOD             | |0        | 00270000   |
B+43     0    |LOC             | |3        | 00020003   |
B+44     0    |LOD             | |0        | 00270000   |
B+45     0    |MUL             | |0        | 00140000   |
B+46     0    |STO             | |0        | 00280000   |
B+47     0    |DEL             | |0        | 002A0000   |
B+48     0    |XIT             | |68       | 00040044   |
B+49     0    |BRS             | |0        | 002E0000   |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 9 IS:
 C(117:118,*,119)  C(114:120,*,121)   A(1,*,121)   A(3,*,120)   A(2,114,114)

 CONTROL AT 68, FROM BLK 8 TO BLK 9 (PASS 1)
C+68     0    |ENT             |C|153      | 00038099   |
B+69     0    |LOC             | |5        | 00020005   |
B+70     0    |LOC             | |2        | 00020002   |
B+71     0    |LOD             | |0        | 00270000   |
B+72     0    |STO             | |0        | 00280000   |
B+73     0    |DEL             | |0        | 002A0000   |
B+74     0    |LOC             | |8        | 00020008   |
B+75     0    |LOC             | |5        | 00020005   |
```

```
B+76   0    |LOD            |  |0        | 00270000  |
B+77   0    |LOC            |  |3        | 00020003  |
B+78   0    |LOD            |  |0        | 00270000  |
B+79   0    |MUL            |  |0        | 00140000  |
B+80   0    |STO            |  |0        | 00280000  |
B+81   0    |DEL            |  |0        | 002A0000  |
B+82   0    |XIT            |  |97       | 00040061  |
B+83   0    |BRS            |  |0        | 002E0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 10 IS:
 C(114:120,*,121)  C(117:118,*,119)  A(8,*,121)  A(5,114,114)  A(1,*,121)
 A(3,*,120)  A(2,114,114)

 CONTROL AT 97, FROM BLK 9 TO BLK 10 (PASS 1)
C+97   0    |ENT            |C |158      | 0003809E  |
B+98   0    |LOC            |  |5        | 00020005  |
B+99   0    |LOC            |  |3        | 00020003  |
B+100  0    |LOD            |  |0        | 00270000  |
B+101  0    |STO            |  |0        | 00280000  |
B+102  0    |DEL            |  |0        | 002A0000  |
B+103  0    |LOC            |  |9        | 00020009  |
B+104  C    |LOC            |  |2        | 00020002  |
B+105  0    |LOD            |  |0        | 00270000  |
B+106  0    |LOC            |  |3        | 00020003  |
B+107  0    |LOD            |  |0        | 00270000  |
B+108  0    |MUL            |  |0        | 00140000  |
B+109  0    |STO            |  |0        | 00280000  |
B+110  0    |DEL            |  |0        | 002A0000  |
B+111  0    |XIT            |  |8        | 00040008  |
B+112  0    |BRS            |  |0        | 002E0000  |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 10 IS:
 C(117:118,*,119)  C(114:120,*,121)  A(9,*,121)  A(5,*,120)  A(8,*,121)
 A(1,*,121)  A(3,*,120)  A(2,114,114)


 CURRENT POOL FOR BLOCK# 2 IS:
 A(2,*,118)  A(1,*,117)

AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

 CONTROL AT 50, FROM BLK 6 TO BLK 7 (PASS 1)
C+50   0    |ENT            |C |143      | 0003608F  |
B+51   0    |LOC            |  |2        | 00020002  |
B+52   0    |INT            |  |2        | 00050002  |
B+53   0    |STO            |  |0        | 00280000  |
B+54   0    |DEL            |  |0        | 002A0000  |
B+55   0    |LOC            |  |4        | 00020004  |
B+56   0    |LOC            |  |2        | 00020002  |
B+57   0    |LOD            |  |0        | 00270000  |
B+58   0    |STO            |  |0        | 00280000  |
B+59   0    |DEL            |  |0        | 002A0000  |
B+60   0    |LOC            |  |6        | 00020006  |
B+61   0    |LOC            |  |4        | 00020004  |
B+62   0    |LOD            |  |0        | 00270000  |
B+63   0    |LOC            |  |3        | 00020003  |
B+64   0    |LOD            |  |0        | 00270000  |
B+65   0    |MUL            |  |0        | 00140000  |
B+66   0    |STO            |  |0        | 00280000  |
B+67   0    |DEL            |  |0        | 002A0000  |
B+68   0    |ENT            |C |153      | 00038099  |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 7 IS:
 C(117:118,*,119)  C(115:124,*,125)  A(3,*,124)  A(6,*,125)  A(4,115,115)
 A(2,115,115)  A(1,*,117)


 CURRENT POOL FOR BLOCK# 9 IS:
 C(117:118,*,119)  C(114:120,*,121)  A(1,*,121)  A(3,*,120)  A(2,114,114)

AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 9 IS:
 A(1,*,126)  A(3,*,127)  A(2,*,128)  C(128:127,*,129)


 CONTROL AT 68, FROM BLK 7 TO BLK 9 (PASS 2)
C+68   0    |ENT            |C |153      | 00038099  |
B+69   0    |LOC            |  |5        | 00020005  |
B+70   0    |LOC            |  |2        | 00020002  |
B+71   0    |LOD            |  |0        | 00270000  |
B+72   0    |STO            |  |0        | 00280000  |
B+73   0    |DEL            |  |0        | 002A0000  |
```

```
B+74   0   LOC      8      00020008
B+75   0   LOC      5      00020005
B+76   0   LOD      0      00270000
B+77   0   LOC      3      00020003
B+78   0   LOD      0      00270000
B+79   0   MUL    C 992    001483E0    COMM SUBEXP ELIM
B+80   0   STO      0      00280000
B+81   0   DEL      0      002A0000
B+82   0   XIT      97     00040061
B+83   0   BRS      0      002E0000

BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 9 IS:
 C(128:127,*,129)  A(8,*,129)  A(5,*,128)  A(1,*,126)  A(3,*,127)  A(2,*,128)

 CURRENT POOL FOR BLOCK# 10 IS:
 C(114:120,*,121)  C(117:118,*,119)  A(8,*,121)  A(5,114,114)  A(1,*,121)
 A(3,*,120)  A(2,114,114)

AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 10 IS:
 A(8,*,130)  A(5,*,131)  A(1,*,132)  A(3,*,133)  A(2,*,131)

 CONTROL AT 97, FROM BLK 9 TO BLK 10 (PASS 2)
C+97   0   ENT    C 158    0003809E
B+98   0   LOC      5      00020005
B+99   0   LOC      3      00020003
B+100  0   LOD      0      00270000
B+101  0   STO      0      00280000
B+102  0   DEL      0      002A0000
B+103  0   LOC      9      00020009
B+104  0   LOC      2      00020002
B+105  0   LOD      0      00270000
B+106  0   LOC      3      00020003
B+107  0   LOD      0      00270000
B+108  0   MUL    C 984    001483D8    COMM SUBEXP ELIM
B+109  0   STO      0      00280000
B+110  0   DEL      0      002A0000
B+111  0   XIT      8      00040008
B+112  0   BRS      0      002E0000

BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 10 IS:
 C(131:133,*,134)  A(9,*,134)  A(5,*,133)  A(8,*,130)  A(1,*,132)  A(3,*,133)
 A(2,*,131)

 CURRENT POOL FOR BLOCK# 2 IS:
 A(2,*,122)  A(1,*,123)

AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

 CONTROL AT 84, FROM BLK 4 TO BLK 5 (PASS 1)
C+84   0   ENT    C 133    00038085
B+85   0   LOC      2      00020002
B+86   0   INT      3      00050003
B+87   0   STO      0      00280000
B+88   0   DEL      0      002A0000
B+89   0   LOC      10     0002000A
B+90   0   LOC      3      00020003
B+91   0   LOD      0      00270000
B+92   0   LOC      2      00020002
B+93   0   LOD      0      00270000
B+94   0   MUL      0      00140000
B+95   0   STO      0      00280000
B+96   0   DEL      0      002A0000
B+97   0   ENT    C 158    0003809E

BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 5 IS:
 C(117:118,*,119)  C(137:116,*,138)  A(3,*,137)  A(10,*,138)  A(2,116,116)
 A(1,*,117)

 CURRENT POOL FOR BLOCK# 10 IS:
 A(8,*,130)  A(5,*,131)  A(1,*,132)  A(3,*,133)  A(2,*,131)

AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 10 IS:
 A(1,*,139)  A(3,*,140)  A(2,*,141)
```

```
CONTROL AT 97, FROM BLK 5 TO BLK 10 (PASS 3)
C+97    0    |ENT          |C|158      | 0003809E  |
B+98    0    |LOC          | |5        | 00020005  |
B+99    0    |LOC          | |3        | 00020003  |
B+100   0    |LOD          | |0        | 00270000  |
B+101   0    |STO          | |0        | 00280000  |
B+102   0    |DEL          | |0        | 002A0000  |
B+103   0    |LOC          | |9        | 00020009  |
B+104   0    |LOC          | |2        | 00020002  |
B+105   0    |LOD          | |0        | 00270000  |
B+106   0    |LOC          | |3        | 00020003  |
B+107   0    |LOD          | |0        | 00270000  |
B+108   0    |MUL          |C|984      | 001483D8  |
B+109   0    |STO          | |0        | 00280000  |
B+110   0    |DEL          | |0        | 002A0000  |
B+111   0    |XIT          | |8        | 00040008  |
B+112   0    |BRS          | |0        | 002E0000  |

BEFORE THE MEET OPERATION:

  OUTPUT POOL FROM BLOCK# 10 IS:
  C(141:140,*,142)  A(9,*,142)  A(5,*,140)  A(1,*,139)  A(3,*,140)  A(2,*,141)

  CURRENT POOL FOR BLOCK# 2 IS:
  A(2,*,135)  A(1,*,136)

AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

  CONTROL AT 113, FROM BLK 2 TO BLK 3 (PASS 1)
C+113   0    |ENT          |C|123      | 0003807B  |

                         FINAL OPTIMIZATION RESULTS
################################################################################
--------------------------------------------------------------------------------

OPTIMIZATION AT 22
         (117,118|13,119)     COMM SUBEXP ELIM

OPTIMIZATION AT 31
         (117,118|13,119)     COMM SUBEXP ELIM

OPTIMIZATION AT 79
         (127,128|45,129)     COMM SUBEXP ELIM

  ****  FORMATTED INTERMEDIATE CODE DUMP  ****
 LOC OFF    OP CODE     CNS ADR     RAW CODE        OPTIMIZATION
      SET                /ETC                            TYPE
***********************************************************************
0      0     |ENT          | |114      | 00030072  |
1      0     |TOGGLE       | |257      | 000B0101  |
2      0     |LOC          | |1        | 00020001  |
3      0     |INT          | |0        | 00050000  |
4      0     |STO          | |0        | 00290000  |
5      0     |LOC          | |2        | 00020002  |
6      0     |INT          | |0        | 00050000  |
7      0     |STO          | |0        | 00290000  |
8      0     |ENT          | |118      | 00030076  |
9      0     |LOC          | |1        | 00020001  |
10     0     |LOD          | |0        | 00270000  |
11     0     |LOC          | |2        | 00020002  |
12     0     |LOD          | |0        | 00270000  |
13     0     |GTR          | |1024     | 00210400  |
14     0     |XIT          | |113      | 00040071  |
15     0     |XIT          | |17       | 00040011  |
16     0     |BSC          | |0        | 002F0000  |
17     0     |ENT          | |128      | 00030080  |
18     0     |LOC          | |1        | 00020001  |
19     0     |LOD          | |0        | 00270000  |
20     0     |LOC          | |2        | 00020002  |
21     0     |LOD          | |0        | 00270000  |
22     0     |GTR          |C|1016     | 002183F8  |     COMM SUBEXP ELIM
23     0     |XIT          | |84       | 00040054  |
24     0     |XIT          | |26       | 0004001A  |
25     0     |BSC          | |0        | 002F0000  |
26     0     |ENT          | |138      | 0003008A  |
27     0     |LOC          | |1        | 00020001  |
28     0     |LOD          | |0        | 00270000  |
29     0     |LOC          | |2        | 00020002  |
30     0     |LOD          | |0        | 00270000  |
31     0     |GTR          |C|1008     | 002183F0  |     COMM SUBEXP ELIM
32     0     |XIT          | |50       | 00040032  |
```

33	0	XIT		35	00040023	
34	0	BSC.		0	002F0000	
35	0	ENT		148	00030094	
36	0	LOC		2	00020002	
37	0	INT		1	00050001	
38	0	STO		0	00280000	
39	0	DEL		0	002A0000	
40	0	LOC		1	00020001	
41	0	LOC		2	00020002	
42	0	LOD		0	00270000	
43	0	LOC		3	00020003	
44	0	LOD		0	00270000	
45	0	MUL		1000	001403E8	
46	0	STO		0	00280000	
47	0	DEL		0	002A0000	
48	0	XIT		68	00040044	
49	0	BRS		0	002E0000	
50	0	ENT		143	0003008F	
51	0	LOC		2	00020002	
52	0	INT		2	00050002	
53	0	STO		0	00280000	
54	0	DEL		0	002A0000	
55	0	LOC		4	00020004	
56	0	LOC		2	00020002	
57	0	LOD		0	00270000	
58	0	STO		0	00280000	
59	0	DEL		0	002A0000	
60	0	LOC		6	00020006	
61	0	LOC		4	00020004	
62	0	LOD		0	00270000	
63	0	LOC		3	00020003	
64	0	LOD		0	00270000	
65	0	MUL		980	001403D4	
66	0	STO		0	00280000	
67	0	DEL		0	002A0000	
68	0	ENT		153	00030099	
69	0	LOC		5	00020005	
70	0	LOC		2	00020002	
71	0	LOD		0	00270000	
72	0	STO		0	00280000	
73	0	DEL		0	002A0000	
74	0	LOC		8	00020008	
75	0	LOC		5	00020005	
76	0	LOD		0	00270000	
77	0	LOC		3	00020003	
78	0	LOD		0	00270000	
79	0	MUL	C	992	001483E0	COMM SUBEXP ELIM
80	0	STO		0	00280000	
81	0	DEL		0	002A0000	
82	0	XIT		97	00040061	
83	0	BRS		0	002E0000	
84	0	ENT		133	00030085	
85	0	LOC		2	00020002	
86	0	INT		3	00050003	
87	0	STO		0	00280000	
88	0	DEL		0	002A0000	
89	0	LOC		10	0002000A	
90	0	LOC		3	00020003	
91	0	LOD		0	00270000	
92	0	LOC		2	00020002	
93	0	LOD		0	00270000	
94	0	MUL		976	001403D0	
95	0	STO		0	00280000	
96	0	DEL		0	002A0000	
97	0	ENT		158	0003009E	
98	0	LOC		5	00020005	
99	0	LOC		3	00020003	
100	0	LOD		0	00270000	
101	0	STO		0	00280000	
102	0	DEL		0	002A0000	
103	0	LOC		9	00020009	
104	0	LOC		2	00020002	
105	0	LOD		0	00270000	
106	0	LOC		3	00020003	
107	0	LOD		0	00270000	
108	0	MUL		984	001403D8	
109	0	STO		0	00280000	
110	0	DEL		0	002A0000	
111	0	XIT		8	00040008	
112	0	BRS		0	002E0000	
113	0	ENT		123	0003007B	

****

FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

```
FINAL POCL FOR  BLOCK# 1 IS:
(NULL)


FINAL POCL FOR  BLOCK# 2 IS:
A(2,*,143)  A(1,*,144)


FINAL POCL FOR  BLOCK# 3 IS:
C(117:118,*,119)  A(2,*,118)  A(1,*,117)


FINAL POCL FOR  BLOCK# 4 IS:
C(117:118,*,119)  A(2,*,118)  A(1,*,117)


FINAL POOL FOR  BLOCK# 5 IS:
C(117:118,*,119)  A(2,*,118)  A(1,*,117)


FINAL POCL FOR  BLOCK# 6 IS:
C(117:118,*,119)  A(2,*,118)  A(1,*,117)


FINAL POCL FOR  BLOCK# 7 IS:
C(117:118,*,119)  A(2,*,118)  A(1,*,117)


FINAL POCL FOR  BLOCK# 8 IS:
C(117:118,*,119)  A(2,*,118)  A(1,*,117)


FINAL POCL FOR  BLOCK# 9 IS:
A(1,*,126)  A(3,*,127)  A(2,*,128)  C(128:127,*,129)


FINAL POCL FOR  BLOCK# 10 IS:
A(1,*,139)  A(3,*,140)  A(2,*,141)

     BASIC BLOCK #1
           BEGINNING AT 0, ENDING AT 8
           BLOCK TRAVERSED 1 TIMES
           0 REFERENCES:
     BASIC BLOCK #2
           BEGINNING AT 8, ENDING AT 16
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
               1
     BASIC BLOCK #3
           BEGINNING AT 113, ENDING AT 113
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
               2
     BASIC BLOCK #4
           BEGINNING AT 17, ENDING AT 25
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
               2
     BASIC BLOCK #5
           BEGINNING AT 84, ENDING AT 97
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
               4
     BASIC BLOCK #6
           BEGINNING AT 26, ENDING AT 34
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
               4
     BASIC BLOCK #7
           BEGINNING AT 50, ENDING AT 68
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
               6
     BASIC BLOCK #8
           BEGINNING AT 35, ENDING AT 49
```

```
         BLOCK TRAVERSED 1 TIMES
         1 REFERENCES:
            6
    BASIC BLOCK #9
         BEGINNING AT 68, ENDING AT 83
         BLOCK TRAVERSED 2 TIMES
         1 REFERENCES:
            8
    BASIC BLOCK #10
         BEGINNING AT 97, ENDING AT 112
         BLOCK TRAVERSED 3 TIMES
         1 REFERENCES:
            9

BLOCK SUMMARY DATA:

BLOCK NUMBER                          NUMBER OF PASSES

    1                                        1
    2                                        1
    3                                        1
    4                                        1
    5                                        1
    6                                        1
    7                                        1
    8                                        1
    9                                        2
    10                                       3

TOTAL NUMBER OF BLOCKS PROCESSED WAS 13
USING THE MAXIMUM CURRENT POOL  BLOCK SELECTION ALGORITHM.

TIME FOR THE OPTIMIZATION WAS 0.268 SECONDS.

                  ***CODE SYNTHESIS FILTER IS COMPLETE***
```

```
CARD |BL|  SYL |
   1 | 0|    0 |COMMENT     TEST PROGRAM # 7          ;
$EXECUTE
   3 | 0|    1 |BEGIN LOCAL    A,B,C,D;
   4 | 1|    1 |A := 1;
   5 | 1|    5 |B := 2;
   6 | 1|    9 |C := 3;
   7 | 1|   13 |A := A * 5;
   8 | 1|   20 |A := A + B;
   9 | 1|   28 |B := B + C;
  10 | 1|   36 |B := B + 1;
  11 | 1|   43 |C := A * 3;
  12 | 1|   50 |C := C + 6;
  13 | 1|   57 |D := 10;
  14 | 1|   61 |IF A GTR B THEN
  15 | 1|   66 |      BEGIN
  16 | 1|   70 |            D := B * C;
  17 | 2|   78 |      END;
  18 | 1|   79 |A := C * B;
  19 | 1|   87 |END
  20 | 1|   87 |EOF
CODE FILE COPIED (87 WORDS)
CONSTANT TABLE COPIED (12 WORDS)
2 RECORDS WRITTEN INTO FILE 1
END OF COMPILATION FEBRUARY 12, 1975.  CLOCK TIME = 20:45:50.31.

20 CARDS WERE READ.
NO ERRORS WERE DETECTED.
```

```
***CODE SYNTHESIS FILTER***

BLOCK SELECTION METHOD WILL BE LAST-IN-FIRST-OUT

***C O M P L E T E   T A B L E   D U M P***
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

***T A B L E   D U M P*** AT LOCATION = 0

CONSTANTS <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
CONSYM    |1   |2   |3   |5   |6   |10
CONTYPE   |INT |INT |INT |INT |INT |INT
CONVAL    |88  |89  |90  |91  |92  |93
CONINT    |1   |2   |3   |5   |6   |10

ADDR TABL <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

VALUE STACK (TOP AT 127)


CONT STK  <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
BLOCK#    |0
ENTRY     |0

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)


  ****  FORMATTED INTERMEDIATE CODE DUMP  ****
  LOC OFF     OP CODE     CNS ADR      RAW CODE       OPTIMIZATION
      SET                     /ETC                        TYPE
*****************************************************************************
  0     0    |ENT          |C|88      | 00038058 |
  1     0    |TOGGLE       | |257     | 000B0101 |
  2     0    |LOC          | |1       | 00020001 |
  3     0    |INT          | |1       | 00050001 |
  4     0    |STO          | |0       | 00280000 |
  5     0    |DEL          | |0       | 002A0000 |
  6     0    |LOC          | |2       | 00020002 |
  7     0    |INT          | |2       | 00050002 |
  8     0    |STO          | |0       | 00280000 |
  9     0    |DEL          | |0       | 002A0000 |
  10    0    |LOC          | |3       | 00020003 |
  11    0    |INT          | |3       | 00050003 |
  12    0    |STO          | |0       | 00280000 |
  13    0    |DEL          | |0       | 002A0000 |
  14    0    |LOC          | |1       | 00020001 |
  15    0    |LOC          | |1       | 00020001 |
  16    0    |LOD          | |0       | 00270000 |
  17    0    |INT          | |4       | 00050004 |
  18    0    |MUL          | |0       | 00140000 |
  19    0    |STO          | |0       | 00280000 |
  20    0    |DEL          | |0       | 002A0000 |
  21    0    |LOC          | |1       | 00020001 |
  22    0    |LOC          | |1       | 00020001 |
  23    0    |LOD          | |0       | 00270000 |
  24    0    |LOC          | |2       | 00020002 |
  25    0    |LOD          | |0       | 00270000 |
  26    0    |ADD          | |0       | 00100000 |
  27    0    |STO          | |0       | 00280000 |
  28    0    |DEL          | |0       | 002A0000 |
  29    0    |LOC          | |2       | 00020002 |
  30    0    |LOC          | |2       | 00020002 |
  31    0    |LOD          | |0       | 00270000 |
  32    0    |LOC          | |3       | 00020003 |
  33    0    |LOD          | |0       | 00270000 |
  34    0    |ADD          | |0       | 00100000 |
  35    0    |STO          | |0       | 00280000 |
  36    0    |DEL          | |0       | 002A0000 |
  37    0    |LOC          | |2       | 00020002 |
  38    0    |LOC          | |2       | 00020002 |
  39    0    |LOD          | |0       | 00270000 |
  40    0    |INT          | |1       | 00050001 |
  41    0    |ADD          | |0       | 00100000 |
```

42	0	STO	0	00280000
43	0	DEL	0	002A0000
44	0	LOC	3	00020003
45	0	LOC	1	00020001
46	0	LOD	0	00270000
47	0	INT	3	00050003
48	0	MUL	0	00140000
49	0	STO	0	00280000
50	0	DEL	0	002A0000
51	0	LOC	3	00020003
52	0	LOC	3	00020003
53	0	LOD	0	00270000
54	0	INT	5	00050005
55	0	ADD	0	00100000
56	0	STO	0	00280000
57	0	DEL	0	002A0000
58	0	LOC	4	00020004
59	0	INT	6	00050006
60	0	STO	0	00280000
61	0	DEL	0	002A0000
62	0	LOC	1	00020001
63	0	LOD	0	00270000
64	0	LOC	2	00020002
65	0	LOD	0	00270000
66	0	GTR	0	00210000
67	0	XIT	79	0004004F
68	0	XIT	70	00040046
69	0	BSC	0	002F0000
70	0	ENT	1	00030001
71	0	LOC	4	00020004
72	0	LOC	2	00020002
73	0	LOD	0	00270000
74	0	LOC	3	00020003
75	0	LOD	0	00270000
76	0	MUL	0	00140000
77	0	STO	0	00280000
78	0	DEL	0	002A0000
79	0	ENT	1	00030001
80	0	LOC	1	00020001
81	0	LOC	3	00020003
82	0	LOD	0	00270000
83	0	LOC	2	00020002
84	0	LOD	0	00270000
85	0	MUL	0	00140000
86	0	STO	0	00280000
87	0	DEL	0	002A0000

```
*************************************************************************************

      BASIC BLOCK #1
            BEGINNING AT 0, ENDING AT 0
            BLOCK TRAVERSED 0 TIMES
            0 REFERENCES:

***E N D   O F   C O M P L E T E   T A B L E   D U M P***
```

B+2	0	LOC	1	00020001
B+3	0	INT	1	00050001
B+4	0	STO	0	00280000
B+5	0	DEL	0	002A0000
B+6	0	LOC	2	00020002
B+7	0	INT	2	00050002
B+8	0	STO	0	00280000
B+9	0	DEL	0	002A0000
B+10	0	LOC	3	00020003
B+11	0	INT	3	00050003
B+12	0	STO	0	00280000
B+13	0	DEL	0	002A0000
B+14	0	LOC	1	00020001
B+15	0	LOC	1	00020001
B+16	0	LOD	0	00270000
B+17	0	INT	4	00050004
B+18	0	MUL	0	00140000
B+19	0	STO	0	00280000
B+20	0	DEL	0	002A0000
B+21	0	LOC	1	00020001
B+22	0	LOC	1	00020001
B+23	0	LOD	0	00270000
B+24	0	LOC	2	00020002
B+25	0	LOD	0	00270000
B+26	0	ADD	0	00100000
B+27	0	STO	0	00280000
B+28	0	DEL	0	002A0000
B+29	0	LOC	2	00020002
B+30	0	LOC	2	00020002
B+31	0	LOD	0	00270000
B+32	0	LOC	3	00020003
B+33	0	LOD	0	00270000

```
B+34    0    |ADD           | |0     | 00100000  |
B+35    0    |STO           | |0     | 00280000  |
B+36    0    |DEL           | |0     | 002A0000  |
B+37    0    |LOC           | |2     | 00020002  |
B+38    0    |LOC           | |2     | 00020002  |
B+39    0    |LOD           | |0     | 00270000  |
B+40    0    |INT           | |1     | 00050001  |
B+41    0    |ADD           | |0     | 00100000  |
B+42    0    |STO           | |0     | 00280000  |
B+43    0    |DEL           | |0     | 002A0000  |
B+44    0    |LOC           | |3     | 00020003  |
B+45    0    |LOC           | |1     | 00020001  |
B+46    0    |LOD           | |0     | 00270000  |
B+47    0    |INT           | |3     | 00050003  |
B+48    0    |MUL           | |0     | 00140000  |
B+49    0    |STO           | |0     | 00280000  |
B+50    0    |DEL           | |0     | 002A0000  |
B+51    0    |LOC           | |3     | 00020003  |
B+52    0    |LOC           | |3     | 00020003  |
B+53    0    |LOD           | |0     | 00270000  |
B+54    0    |INT           | |5     | 00050005  |
B+55    0    |ADD           | |0     | 00100000  |
B+56    0    |STO           | |0     | 00280000  |
B+57    0    |DEL           | |0     | 002A0000  |
B+58    0    |LOC           | |4     | 00020004  |
B+59    0    |INT           | |6     | 00050006  |
B+60    0    |STO           | |0     | 00280000  |
B+61    0    |DEL           | |0     | 002A0000  |
B+62    0    |LOC           | |1     | 00020001  |
B+63    0    |LOD           | |0     | 00270000  |
B+64    0    |LOC           | |2     | 00020002  |
B+65    0    |LOD           | |0     | 00270000  |
B+66    0    |GTR           | |0     | 00210000  |
B+67    0    |XIT           | |79    | 0004004F  |
B+68    0    |XIT           | |70    | 00040046  |
B+69    0    |BSC           | |0     | 002F0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 2 IS:
 C(88:91,*,91)  C(91:89,*,96)  C(89:90,*,91)  C(91:88,*,92)  C(96:90,*,100)
 C(100:92,*,102)  C(96:92,*,104)  A(4,93,93)  A(3,102,102)  A(2,92,92)
 A(1,96,96)

 CONTROL AT 70, FROM BLK 1 TO BLK 2 (PASS 1)
C+70    0    |ENT           |C|92    | 0003805C  |
B+71    0    |LOC           | |4     | 00020004  |
B+72    0    |LOC           | |2     | 00020002  |
B+73    0    |LOD           | |0     | 00270000  |
B+74    0    |LOC           | |3     | 00020003  |
B+75    0    |LOD           | |0     | 00270000  |
B+76    0    |MUL           | |0     | 00140000  |
B+77    0    |STO           | |0     | 00280000  |
B+78    0    |DEL           | |0     | 002A0000  |
B+79    0    |ENT           | |1     | 00030001  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 3 IS:
 C(96:92,*,104)  C(100:92,*,102)  C(96:90,*,100)  C(91:88,*,92)  C(89:90,*,91)
 C(91:89,*,96)  C(88:91,*,91)  C(92:102,*,106)  A(4,106,106)  A(3,102,102)
 A(2,92,92)  A(1,96,96)

 CONTROL AT 79, FROM BLK 2 TO BLK 3 (PASS 1)
C+79    0    |ENT           |C|97    | 00038061  |
B+80    0    |LOC           | |1     | 00020001  |
B+81    0    |LOC           | |3     | 00020003  |
B+82    0    |LOD           | |0     | 00270000  |
B+83    0    |LOC           | |2     | 00020002  |
B+84    0    |LOD           | |0     | 00270000  |
B+85    0    |MUL           | |0     | 00140000  |
B+86    0    |STO           | |0     | 00280000  |
B+87    0    |DEL           | |0     | 002A0000  |

                         FINAL OPTIMIZATION RESULTS
###########################################################################
---------------------------------------------------------------------------

OPTIMIZATION AT 18
        (88,91|18,91)          CONSTANT PROPAGATION

OPTIMIZATION AT 26
        (89,91|26,96)          CONSTANT PROPAGATION

OPTIMIZATION AT 34
        (89,90|34,91)          CONSTANT PROPAGATION

OPTIMIZATION AT 41
        (88,91|41,92)          CONSTANT PROPAGATION
```

```
OPTIMIZATION AT 48
          (96,90|48,100)        CONSTANT PROPAGATION

OPTIMIZATION AT 55
          (92,100|55,102)       CONSTANT PROPAGATION

OPTIMIZATION AT 66
          (96,92|66,104)        CONSTANT PROPAGATION

OPTIMIZATION AT 76
          (102,92|76,106)       CONSTANT PROPAGATION

OPTIMIZATION AT 85
          (102,92|76,106)       COMM SUBEXP ELIM & CONS PROP
```

**** FORMATTED INTERMEDIATE CODE DUMP ****

LOC	OFF SET	OP CODE	CNS	ADR /ETC	RAW CODE	OPTIMIZATION TYPE
0	0	ENT		88	00030058	
1	0	TOGGLE		257	00080101	
2	0	LOC		1	00020001	
3	0	INT		1	00050001	
4	0	STO		0	00280000	
5	0	DEL		0	002A0000	
6	0	LOC		2	00020002	
7	0	INT		2	00050002	
8	0	STO		0	00280000	
9	0	DEL		0	002A0000	
10	0	LOC		3	00020003	
11	0	INT		3	00050003	
12	0	STO		0	00280000	
13	0	DEL		0	002A0000	
14	0	LOC		1	00020001	
15	0	LOC		1	00020001	
16	0	LOD		0	00270000	
17	0	INT		4	00050004	
18	0	MUL	C	1024	00148400	CONSTANT PROPAGATION
19	0	STO		0	00280000	
20	0	DEL		0	002A0000	
21	0	LOC		1	00020001	
22	0	LOC		1	00020001	
23	0	LOD		0	00270000	
24	0	LOC		2	00020002	
25	0	LOD		0	00270000	
26	0	ADD	C	1020	001083FC	CONSTANT PROPAGATION
27	0	STO		0	00280000	
28	0	DEL		0	002A0000	
29	0	LOC		2	00020002	
30	0	LOC		2	00020002	
31	0	LOD		0	00270000	
32	0	LOC		3	00020003	
33	0	LOD		0	00270000	
34	0	ADD	C	1016	001083F8	CONSTANT PROPAGATION
35	0	STO		0	00280000	
36	0	DEL		0	002A0000	
37	0	LOC		2	00020002	
38	0	LOC		2	00020002	
39	0	LOD		0	00270000	
40	0	INT		1	00050001	
41	0	ADD	C	1012	001083F4	CONSTANT PROPAGATION
42	0	STO		0	00280000	
43	0	DEL		0	002A0000	
44	0	LOC		3	00020003	
45	0	LOC		1	00020001	
46	0	LOD		0	00270000	
47	0	INT		3	00050003	
48	0	MUL	C	1008	001483F0	CONSTANT PROPAGATION
49	0	STO		0	00280000	
50	0	DEL		0	002A0000	
51	0	LOC		3	00020003	
52	0	LOC		3	00020003	
53	0	LOD		0	00270000	
54	0	INT		5	00050005	
55	0	ADD	C	1004	001083EC	CONSTANT PROPAGATION
56	0	STO		0	00280000	
57	0	DEL		0	002A0000	
58	0	LOC		4	00020004	
59	0	INT		6	00050006	
60	0	STO		0	00280000	
61	0	DEL		0	002A0000	
62	0	LOC		1	00020001	
63	0	LOD		0	00270000	
64	0	LOC		2	00020002	
65	0	LOD		0	00270000	
66	0	GTR	C	1000	002183E8	CONSTANT PROPAGATION
67	0	XIT		79	0004004F	

68	0	XIT		70	00040046	
69	0	BSC		0	002F0000	
70	0	ENT		92	0003005C	
71	0	LOC		4	00020004	
72	0	LOC		2	00020002	
73	0	LOD		0	00270000	
74	0	LOC		3	00020003	
75	0	LOD		0	00270000	
76	0	MUL	C	982	001483D6	CONSTANT PROPAGATION
77	0	STO		0	00280000	
78	0	DEL		0	002A0000	
79	0	ENT		97	00030061	
80	0	LOC		1	00020001	
81	0	LOC		3	00020003	
82	0	LOD		0	00270000	
83	0	LOC		2	00020002	
84	0	LOD		0	00270000	
85	0	MUL	C	962	001483C2	COMM SUBEXP ELIM & CONS PROP
86	0	STO		0	00280000	
87	0	DEL		0	002A0000	

```
****************************************************************************

       FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

 FINAL POOL FOR  BLOCK# 1 IS:
 (NULL)

 FINAL POOL FOR  BLOCK# 2 IS:
 C(88:91,*,91)  C(91:89,*,96)  C(89:90,*,91)  C(91:88,*,92)  C(96:90,*,100)
 C(100:92,*,102)  C(96:92,*,104)  A(4,93,93)  A(3,102,102)  A(2,92,92)
 A(1,96,96)

 FINAL POOL FOR  BLOCK# 3 IS:
 C(96:92,*,104)  C(100:92,*,102)  C(96:90,*,100)  C(91:88,*,92)  C(89:90,*,91)
 C(91:89,*,96)  C(88:91,*,91)  C(92:102,*,106)  A(4,106,106)  A(3,102,102)
 A(2,92,92)  A(1,96,96)

     BASIC BLOCK #1
           BEGINNING AT 0, ENDING AT 69
           BLOCK TRAVERSED 1 TIMES
           0 REFERENCES:
     BASIC BLOCK #2
           BEGINNING AT 70, ENDING AT 79
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
               1
     BASIC BLOCK #3
           BEGINNING AT 79, ENDING AT 87
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
               2

BLOCK SUMMARY DATA:

BLOCK NUMBER                     NUMBER OF PASSES

    1                                   1
    2                                   1
    3                                   1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 3
USING THE LAST-IN-FIRST-OUT  BLOCK SELECTION ALGORITHM.

TIME FOR THE OPTIMIZATION WAS 0.295 SECONDS.

                 ***CODE SYNTHESIS FILTER IS COMPLETE***
```

```
CARD |BL| SYL |
   1 | 0|   0 |COMMENT     TEST PROGRAM # 8          ;
$EXECUTE
   3 | 0|   1 |BEGIN LOCAL    A,B,C,D,E,F,G,H,I;
   4 | 1|   1 |B := 5;
   5 | 1|   5 |C := 10;
   6 | 1|   9 |A := B * C;
   7 | 1|  17 |IF A GTR B THEN
   8 | 1|  22 |      BEGIN
   9 | 1|  26 |           C := 30;
  10 | 2|  30 |           E := C;
  11 | 2|  35 |           G := B * E;
  12 | 2|  43 |      END ELSE
  13 | 1|  43 |      BEGIN
  14 | 1|  46 |      B := B + 2;
  15 | 2|  53 |      D := B;
  16 | 2|  58 |      F := D * C;
  17 | 2|  66 |      END;
  18 | 1|  67 |H := C * B;
  19 | 1|  75 |I := B * C;
  20 | 1|  83 |WHILE H GTR C DO
  21 | 1|  93 |   BEGIN
  22 | 1|  93 |      C := C + 1;
  23 | 2| 100 |      F := 1 + 2;
  24 | 2| 106 |      G := G + 0;
  25 | 2| 113 |      A := A * 1;
  26 | 2| 120 |      B := B * 0;
  27 | 2| 127 |   END;
  28 | 1| 130 |END
  29 | 1| 130 |EOF
CODE FILE COPIED (130 WORDS)
CONSTANT TABLE COPIED (12 WORDS)
2 RECORDS WRITTEN INTO FILE 1
END OF COMPILATION FEBRUARY 12, 1975.  CLOCK TIME = 20:50:41.39.

29 CARDS WERE READ.
NO ERRORS WERE DETECTED.
```

CODE SYNTHESIS FILTER

BLOCK SELECTION METHOD WILL BE STEEPEST DESCENT (MINIMUM CURRENT POOL)

C O M P L E T E   T A B L E   D U M P
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

T A B L E   D U M P AT LOCATION = 0

CONSTANTS	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
CONSYM	5	10	30	2	1	0								
CONTYPE	INT	INT	INT	INT	INT	INT								
CONVAL	131	132	133	134	135	136								
CONINT	5	10	30	2	1	0								

ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
(EMPTY)

VALUE STACK (TOP AT 127)

CONT STK	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
BLOCK#	0													
ENTRY	0													

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
(EMPTY)

**** FORMATTED INTERMEDIATE CODE DUMP ****

LOC	OFF SET	OP CODE	CNS -	ADR /ETC	RAW CODE	OPTIMIZATION TYPE
0	0	ENT	C	131	00038083	
1	0	TOGGLE		257	000B0101	
2	0	LOC		2	00020002	
3	0	INT		1	00050001	
4	0	STO		0	00280000	
5	0	DEL		0	002A0000	
6	0	LOC		3	00020003	
7	0	INT		2	00050002	
8	0	STO		0	00280000	
9	0	DEL		0	002A0000	
10	0	LOC		1	00020001	
11	0	LOC		2	00020002	
12	0	LOD		0	00270000	
13	0	LOC		3	00020003	
14	0	LOD		0	00270000	
15	0	MUL		0	00140000	
16	0	STO		0	00280000	
17	0	DEL		0	002A0000	
18	0	LOC		1	00020001	
19	0	LOD		0	00270000	
20	0	LOC		2	00020002	
21	0	LOD		0	00270000	
22	0	GTR		0	00210000	
23	0	XIT		46	0004002E	
24	0	XIT		26	0004001A	
25	0	BSC		0	002F0000	
26	0	ENT		1	00030001	
27	0	LOC		3	00020003	
28	0	INT		3	00050003	
29	0	STO		0	00280000	
30	0	DEL		0	002A0000	
31	0	LOC		5	00020005	
32	0	LOC		3	00020003	
33	0	LOD		0	00270000	
34	0	STO		0	00280000	
35	0	DEL		0	002A0000	
36	0	LOC		7	00020007	
37	0	LOC		2	00020002	
38	0	LOD		0	00270000	
39	0	LOC		5	00020005	
40	0	LOD		0	00270000	
41	0	MUL		0	00140000	

42	0	STO	0	00280000
43	0	DEL	0	002A0000
44	0	XIT	67	00040043
45	0	BRS	0	002E0000
46	0	ENT	1	00030001
47	0	LOC	2	00020002
48	0	LOC	2	00020002
49	0	LOD	0	00270000
50	0	INT	4	00050004
51	0	ADD	0	00100000
52	0	STO	0	00280000
53	0	DEL	0	002A0000
54	0	LOC	4	00020004
55	0	LOC	2	00020002
56	0	LOD	0	00270000
57	0	STO	0	00280000
58	0	DEL	0	002A0000
59	0	LOC	6	00020006
60	0	LOC	4	00020004
61	0	LOD	0	00270000
62	0	LOC	3	00020003
63	0	LOD	0	00270000
64	0	MUL	0	00140000
65	0	STO	0	00280000
66	0	DEL	0	002A0000
67	0	ENT	1	00030001
68	0	LOC	8	00020008
69	0	LOC	3	00020003
70	0	LOD	0	00270000
71	0	LOC	2	00020002
72	0	LOD	0	00270000
73	0	MUL	0	00140000
74	0	STO	0	00280000
75	0	DEL	0	002A0000
76	0	LOC	9	00020009
77	0	LOC	2	00020002
78	0	LOD	0	00270000
79	0	LOC	3	00020003
80	0	LOD	0	00270000
81	0	MUL	0	00140000
82	0	STO	0	00280000
83	0	DEL	0	002A0000
84	0	ENT	1	00030001
85	0	LOC	8	00020008
86	0	LOD	0	00270000
87	0	LOC	3	00020003
88	0	LOD	0	00270000
89	0	GTR	0	00210000
90	0	XIT	130	00040082
91	0	XIT	93	0004005D
92	0	BSC	0	002F0000
93	0	ENT	1	00030001
94	0	LOC	3	00020003
95	0	LOC	3	00020003
96	0	LOD	0	00270000
97	0	INT	5	00050005
98	0	ADD	0	00100000
99	0	STO	0	00280000
100	0	DEL	0	002A0000
101	0	LOC	6	00020006
102	0	INT	5	00050005
103	0	INT	4	00050004
104	0	ADD	0	00100000
105	0	STO	0	00280000
106	0	DEL	0	002A0000
107	0	LOC	7	00020007
108	0	LOC	7	00020007
109	0	LOD	0	00270000
110	0	INT	6	00050006
111	0	ADD	0	00100000
112	0	STO	0	00280000
113	0	DEL	0	002A0000
114	0	LOC	1	00020001
115	0	LOC	1	00020001
116	0	LOD	0	00270000
117	0	INT	5	00050005
118	0	MUL	0	00140000
119	0	STO	0	00280000
120	0	DEL	0	002A0000
121	0	LOC	2	00020002
122	0	LOC	2	00020002
123	0	LOD	0	00270000
124	0	INT	6	00050006
125	0	MUL	0	00140000
126	0	STO	0	00280000
127	0	DEL	0	002A0000
128	0	XIT	84	00040054
129	0	BRS	0	002E0000

```
130   0   |ENT          | |1      | 00030001  |
**************************************************************************************************

      BASIC BLOCK #1
            BEGINNING AT 0, ENDING AT 0
            BLOCK TRAVERSED 0 TIMES
            0 REFERENCES:

***E N D    O F    C O M P L E T E    T A B L E    D U M P***

B+2     0   |LOC          | |2      | 00020002  |
B+3     0   |INT          | |1      | 00050001  |
B+4     0   |STO          | |0      | 00280000  |
B+5     0   |DEL          | |0      | 002A0000  |
B+6     0   |LOC          | |3      | 00020003  |
B+7     0   |INT          | |2      | 00050002  |
B+8     0   |STO          | |0      | 00280000  |
B+9     0   |DEL          | |0      | 002A0000  |
B+10    0   |LOC          | |1      | 00020001  |
B+11    0   |LOC          | |2      | 00020002  |
B+12    0   |LOD          | |0      | 00270000  |
B+13    0   |LOC          | |3      | 00020003  |
B+14    0   |LOD          | |0      | 00270000  |
B+15    0   |MUL          | |0      | 00140000  |
B+16    0   |STO          | |0      | 00280000  |
B+17    0   |DEL          | |0      | 002A0000  |
B+18    0   |LOC          | |1      | 00020001  |
B+19    0   |LOD          | |0      | 00270000  |
B+20    0   |LOC          | |2      | 00020002  |
B+21    0   |LOD          | |0      | 00270000  |
B+22    0   |GTR          | |0      | 00210000  |
B+23    0   |XIT          | |46     | 0004002E  |
B+24    0   |XIT          | |26     | 0004001A  |
B+25    0   |BSC          | |0      | 002F0000  |

  INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 2 IS:
 C(131:132,*,138)  C(138:131,*,140)  A(1,138,138)  A(3,132,132)  A(2,131,131)

  CONTROL AT 26, FROM BLK 1 TO BLK 2 (PASS 1)
C+26    0   |ENT          |C|135    | 00038087  |
B+27    0   |LOC          | |3      | 00020003  |
B+28    0   |INT          | |3      | 00050003  |
B+29    0   |STO          | |0      | 00280000  |
B+30    0   |DEL          | |0      | 002A0000  |
B+31    0   |LOC          | |5      | 00020005  |
B+32    0   |LOC          | |3      | 00020003  |
B+33    0   |LOD          | |0      | 00270000  |
B+34    0   |STO          | |0      | 00280000  |
B+35    0   |DEL          | |0      | 002A0000  |
B+36    0   |LOC          | |7      | 00020007  |
B+37    0   |LOC          | |2      | 00020002  |
B+38    0   |LOD          | |0      | 00270000  |
B+39    0   |LOC          | |5      | 00020005  |
B+40    0   |LOD          | |0      | 00270000  |
B+41    0   |MUL          | |0      | 00140000  |
B+42    0   |STO          | |0      | 00280000  |
B+43    0   |DEL          | |0      | 002A0000  |
B+44    0   |XIT          | |67     | 00040043  |
B+45    0   |BRS          | |0      | 002E0000  |

  INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 3 IS:
  C(138:131,*,140)  C(131:132,*,138)  C(131:133,*,142)  A(7,142,142)
  A(5,133,133)  A(3,133,133)  A(1,138,138)  A(2,131,131)

  CONTROL AT 67, FROM BLK 2 TO BLK 3 (PASS 1)
C+67    0   |ENT          |C|140    | 0003808C  |
B+68    0   |LOC          | |8      | 00020008  |
B+69    0   |LOC          | |3      | 00020003  |
B+70    0   |LOD          | |0      | 00270000  |
B+71    0   |LOC          | |2      | 00020002  |
B+72    0   |LOD          | |0      | 00270000  |
B+73    0   |MUL          | |0      | 00140000  |
B+74    0   |STO          | |0      | 00280000  |
B+75    0   |DEL          | |0      | 002A0000  |
B+76    0   |LOC          | |9      | 00020009  |
B+77    0   |LOC          | |2      | 00020002  |
B+78    0   |LOD          | |0      | 00270000  |
B+79    0   |LOC          | |3      | 00020003  |
B+80    0   |LOD          | |0      | 00270000  |
B+81    0   |MUL          | |0      | 00140000  |
B+82    0   |STO          | |0      | 00280000  |
B+83    0   |DEL          | |0      | 002A0000  |
B+84    0   |ENT          | |1      | 00030001  |

  INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 4 IS:
  C(131:133,*,142)  C(131:132,*,138)  C(138:131,*,140)  A(9,142,142)
  A(8,142,142)  A(7,142,142)  A(5,133,133)  A(3,133,133)  A(1,138,138)
```

```
A(2,131,131)

 CONTROL AT 84, FROM BLK 3 TO BLK 4 (PASS 1)
C+84    0   |ENT        |C|145     | 00038091 |
B+85    0   |LOC        | |8       | 00020008 |
B+86    0   |LOD        | |0       | 00270000 |
B+87    0   |LOC        | |3       | 00020003 |
B+88    0   |LOD        | |0       | 00270000 |
B+89    0   |GTR        | |0       | 00210000 |
B+90    0   |XIT        | |130     | 00040082 |
B+91    0   |XIT        | |93      | 0004005D |
B+92    0   |BSC        | |0       | 002F0000 |

 INITIAL CURRENT OPTIMIZED PCOL TO BE USED FOR  BLOCK# 5 IS:
 C(138:131,*,140)  C(131:132,*,138)  C(131:133,*,142)  C(142:133,*,140)
 A(9,142,142)  A(8,142,142)  A(7,142,142)  A(5,133,133)  A(3,133,133)
 A(1,138,138)  A(2,131,131)

 CONTROL AT 93, FROM BLK 4 TO BLK 5 (PASS 1)
C+93    0   |ENT        |C|150     | 00038096 |
B+94    0   |LOC        | |3       | 00020003 |
B+95    0   |LOC        | |3       | 00020003 |
B+96    0   |LOD        | |0       | 00270000 |
B+97    0   |INT        | |5       | 00050005 |
B+98    0   |ADD        | |0       | 00100000 |
B+99    0   |STO        | |0       | 00280000 |
B+100   0   |DEL        | |0       | 002A0000 |
B+101   0   |LOC        | |6       | 00020006 |
B+102   0   |INT        | |5       | 00050005 |
B+103   0   |INT        | |4       | 00050004 |
B+104   0   |ADD        | |0       | 00100000 |
B+105   0   |STO        | |0       | 00280000 |
B+106   0   |DEL        | |0       | 002A0000 |
B+107   0   |LOC        | |7       | 00020007 |
B+108   0   |LOC        | |7       | 00020007 |
B+109   0   |LOD        | |0       | 00270000 |
B+110   0   |INT        | |6       | 00050006 |
B+111   0   |ADD        | |0       | 00100000 |
B+112   0   |STO        | |0       | 00280000 |
B+113   0   |DEL        | |0       | 002A0000 |
B+114   0   |LOC        | |1       | 00020001 |
B+115   0   |LOC        | |1       | 00020001 |
B+116   0   |LOD        | |0       | 00270000 |
B+117   0   |INT        | |5       | 00050005 |
B+118   0   |MUL        | |0       | 00140000 |
B+119   0   |STO        | |0       | 00280000 |
B+120   0   |DEL        | |0       | 002A0000 |
B+121   0   |LOC        | |2       | 00020002 |
B+122   0   |LOC        | |2       | 00020002 |
B+123   0   |LOD        | |0       | 00270000 |
B+124   0   |INT        | |6       | 00050006 |
B+125   0   |MUL        | |0       | 00140000 |
B+126   0   |STO        | |0       | 00280000 |
B+127   0   |DEL        | |0       | 002A0000 |
B+128   0   |XIT        | |84      | 00040054 |
B+129   0   |BRS        | |0       | 002E0000 |

BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 5 IS:
 C(142:133,*,140)  C(131:133,*,142)  C(131:132,*,138)  C(138:131,*,140)
 C(133:135,*,145)  C(131:133,*,145)  C(135:134,*,147)  C(142:136,*,142)
 C(138:135,*,138)  C(131:136,*,136)  A(2,136,136)  A(1,138,138)  A(7,142,142)
 A(6,147,147)  A(3,145,145)  A(9,142,142)  A(8,142,142)  A(5,133,133)

 CURRENT POOL FOR BLOCK# 4 IS:
 C(131:133,*,142)  C(131:132,*,138)  C(138:131,*,140)  A(9,142,142)
 A(8,142,142)  A(7,142,142)  A(5,133,133)  A(3,133,133)  A(1,138,138)
 A(2,131,131)

AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT PCOL FOR  BLOCK# 4 IS:
 A(9,142,142)  A(8,142,142)  A(7,142,142)  A(5,133,133)  A(3,*,151)
 A(1,138,138)  A(2,*,152)

 CONTROL AT 84, FROM BLK 5 TO BLK 4 (PASS 2)
C+84    0   |ENT        |C|145     | 00038091 |
B+85    0   |LOC        | |8       | 00020008 |
B+86    0   |LOD        | |0       | 00270000 |
B+87    0   |LOC        | |3       | 00020003 |
B+88    0   |LOD        | |0       | 00270000 |
B+89    0   |GTR        |C|988     | 002183DC |    CONSTANT PROPAGATION
B+90    0   |XIT        | |130     | 00040082 |
B+91    0   |XIT        | |93      | 0004005D |
B+92    0   |BSC        | |0       | 002F0000 |
```

```
OPTIMIZATION AT 15
        (131,132|15,138)     CONSTANT PROPAGATION

OPTIMIZATION AT 22
        (138,131|22,140)     CONSTANT PROPAGATION

OPTIMIZATION AT 41
        (131,133|41,142)     CONSTANT PROPAGATION

OPTIMIZATION AT 73
        (131,133|41,142)     COMM SUBEXP ELIM & CONS PROP

OPTIMIZATION AT 81
        (131,133|41,142)     COMM SUBEXP ELIM & CONS PROP

OPTIMIZATION AT 104
        (134,135|104,147)    CONSTANT PROPAGATION

OPTIMIZATION AT 111
        (136,142|111,142)    CONSTANT PROPAGATION

OPTIMIZATION AT 118
        (135,138|118,138)    CONSTANT PROPAGATION

OPTIMIZATION AT 125
        (136,155|125,136)    SIMPLIFICATION
```

**** FORMATTED INTERMEDIATE CODE DUMP ****

LOC	OFF SET	OP CODE	CNS	ADR /ETC	RAW CODE	OPTIMIZATION TYPE
0	0	ENT		131	00030083	
1	0	TOGGLE		257	00080101	
2	0	LOC		2	00020002	
3	0	INT		1	00050001	
4	0	STO		0	00280000	
5	0	DEL		0	002A0000	
6	0	LOC		3	00020003	
7	0	INT		2	00050002	
8	0	STO		0	00280000	
9	0	DEL		0	002A0000	
10	0	LOC		1	00020001	
11	0	LOC		2	00020002	
12	0	LOD		0	00270000	
13	0	LOC		3	00020003	
14	0	LOD		0	00270000	
15	0	MUL	C	1024	00148400	CONSTANT PROPAGATION
16	0	STO		0	00280000	
17	0	DEL		0	002A0000	
18	0	LOC		1	00020001	
19	0	LOD		0	00270000	
20	0	LOC		2	00020002	
21	0	LOD		0	00270000	
22	0	GTR	C	1020	002183FC	CONSTANT PROPAGATION
23	0	XIT		46	0004002E	
24	0	XIT		26	0004001A	
25	0	BSC		0	002F0000	
26	0	ENT		135	00030087	
27	0	LOC		3	00020003	
28	0	INT		3	00050003	
29	0	STO		0	00280000	
30	0	DEL		0	002A0000	
31	0	LOC		5	00020005	
32	0	LOC		3	00020003	
33	0	LOD		0	00270000	
34	0	STO		0	00280000	
35	0	DEL		0	002A0000	
36	0	LOC		7	00020007	
37	0	LOC		2	00020002	
38	0	LOD		0	00270000	
39	0	LOC		5	00020005	
40	0	LOD		0	00270000	
41	0	MUL	C	1012	001483F4	CONSTANT PROPAGATION
42	0	STO		0	00280000	
43	0	DEL		0	002A0000	
44	0	XIT		67	00040043	
45	0	BRS		0	002E0000	
46	0	ENT		1	00030001	
47	0	LOC		2	00020002	
48	0	LOC		2	00020002	
49	0	LOD		0	00270000	
50	0	INT		4	00050004	
51	0	ADD		0	00100000	
52	0	STO		0	00280000	
53	0	DEL		0	002A0000	
54	0	LOC		4	00020004	
55	0	LOC		2	00020002	

56	0	LOD		0	00270000	
57	0	STO		0	00280000	
58	0	DEL		0	002A0000	
59	0	LOC		6	00020006	
60	0	LOC		4	00020004	
61	0	LOD		0	00270000	
62	0	LOC		3	00020003	
63	0	LOD		0	00270000	
64	0	MUL		0	00140000	
65	0	STO		0	00280000	
66	0	DEL		0	002A0000	
67	0	ENT		140	0003008C	
68	0	LOC		8	00020008	
69	0	LOC		3	00020003	
70	0	LOD		0	00270000	
71	0	LOC		2	00020002	
72	0	LOD		0	00270000	
73	0	MUL	C	1002	001483EA	COMM SUBEXP ELIM & CONS PROP
74	0	STO		0	00280000	
75	0	DEL		0	002A0000	
76	0	LOC		9	00020009	
77	0	LOC		2	00020002	
78	0	LOD		0	00270000	
79	0	LOC		3	00020003	
80	0	LOD		0	00270000	
81	0	MUL	C	998	001483E6	COMM SUBEXP ELIM & CONS PROP
82	0	STO		0	00280000	
83	0	DEL		0	002A0000	
84	0	ENT		145	00030091	
85	0	LOC		8	00020008	
86	0	LOD		0	00270000	
87	0	LOC		3	00020003	
88	0	LOD		0	00270000	
89	0	GTR		988	002103DC	
90	0	XIT		130	00040082	
91	0	XIT		93	0004005D	
92	0	BSC		0	002F0000	
93	0	ENT		150	00030096	
94	0	LOC		3	00020003	
95	0	LOC		3	00020003	
96	0	LOD		0	00270000	
97	0	INT		5	00050005	
98	0	ADD		976	001003D0	
99	0	STO		0	00280000	
100	0	DEL		0	002A0000	
101	0	LOC		6	00020006	
102	0	INT		5	00050005	
103	0	INT		4	00050004	
104	0	ADD	C	972	001083CC	CONSTANT PROPAGATION
105	0	STO		0	00280000	
106	0	DEL		0	002A0000	
107	0	LOC		7	00020007	
108	0	LOC		7	00020007	
109	0	LOD		0	00270000	
110	0	INT		6	00050006	
111	0	ADD	C	968	001083C8	CONSTANT PROPAGATION
112	0	STO		0	00280000	
113	0	DEL		0	002A0000	
114	0	LOC		1	00020001	
115	0	LOC		1	00020001	
116	0	LOD		0	00270000	
117	0	INT		5	00050005	
118	0	MUL	C	964	001483C4	CONSTANT PROPAGATION
119	0	STO		0	00280000	
120	0	DEL		0	002A0000	
121	0	LOC		2	00020002	
122	0	LOC		2	00020002	
123	0	LOD		0	00270000	
124	0	INT		6	00050006	
125	0	MUL	C	960	001483C0	SIMPLIFICATION
126	0	STO		0	00280000	
127	0	DEL		0	002A0000	
128	0	XIT		84	00040054	
129	0	BRS		0	002E0000	
130	0	ENT		155	0003009B	

****

FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

FINAL POOL FOR BLOCK# 1 IS:
(NULL)

```
FINAL POOL FOR  BLOCK# 2 IS:
C(131:132,*,138)  C(138:131,*,140)  A(1,138,138)  A(3,132,132)  A(2,131,131)


FINAL POOL FOR  BLOCK# 3 IS:
C(138:131,*,140)  C(131:132,*,138)  C(131:133,*,142)  A(7,142,142)
A(5,133,133)  A(3,133,133)  A(1,138,138)  A(2,131,131)


FINAL POOL FOR  BLOCK# 4 IS:
A(9,142,142)  A(8,142,142)  A(7,142,142)  A(5,133,133)  A(3,*,160)
A(1,138,138)  A(2,*,161)


FINAL POOL FOR  BLOCK# 5 IS:
A(9,142,142)  A(8,142,142)  A(7,142,142)  A(5,133,133)  A(3,*,154)
A(1,138,138)  A(2,*,155)  C(142:154,*,156)


FINAL POOL FOR  BLOCK# 6 IS:
C(142:151,*,153)  A(9,142,142)  A(8,142,142)  A(7,142,142)  A(5,133,133)
A(3,*,151)  A(1,138,138)  A(2,*,152)

      BASIC BLOCK #1
            BEGINNING AT 0, ENDING AT 25
            BLOCK TRAVERSED 1 TIMES
            0 REFERENCES:
      BASIC BLOCK #2
            BEGINNING AT 26, ENDING AT 45
            BLOCK TRAVERSED 1 TIMES
            1 REFERENCES:
               1
      BASIC BLOCK #3
            BEGINNING AT 67, ENDING AT 84
            BLOCK TRAVERSED 1 TIMES
            1 REFERENCES:
               2
      BASIC BLOCK #4
            BEGINNING AT 84, ENDING AT 92
            BLOCK TRAVERSED 2 TIMES
            1 REFERENCES:
               3
      BASIC BLOCK #5
            BEGINNING AT 93, ENDING AT 129
            BLOCK TRAVERSED 2 TIMES
            1 REFERENCES:
               4
      BASIC BLOCK #6
            BEGINNING AT 130, ENDING AT 130
            BLOCK TRAVERSED 1 TIMES
            1 REFERENCES:
               4
BLOCK SUMMARY DATA:

BLOCK NUMBER                     NUMBER OF PASSES

     1                                  1
     2                                  1
     3                                  1
     4                                  2
     5                                  2
     6                                  1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 8
USING THE STEEPEST DESCENT (MINIMUM CURRENT POOL)  BLOCK SELECTION ALGORITHM.

TIME FOR THE OPTIMIZATION WAS 0.460 SECONDS.

                  ***CODE SYNTHESIS FILTER IS COMPLETE***
```

```
CARD |BL| SYL |
   1 | 0|   0 |COMMENT      TEST PROGRAM # 9        ;
$EXECUTE
   3 | 0|   1 |BEGIN LOCAL    A,B,C,D,E,F,G,H,I,J,K;
   4 | 1|   1 |READ(A, B, C);
   5 | 1|  10 |I := B * C;
   6 | 1|  18 |     J := C * B;
   7 | 1|  26 |     WHILE A GTR B DO
   8 | 1|  36 |          BEGIN
   9 | 1|  36 |          WHILE A GTR B DO
  10 | 2|  46 |               BEGIN
  11 | 2|  46 |               WHILE A GTR B DO
  12 | 3|  56 |                    B := B + 1;
  13 | 3|  66 |                    J := B * C;
  14 | 3|  74 |                    K := 1;
  15 | 3|  78 |                    G := 1 + 3;
  16 | 3|  84 |               END;
  17 | 2|  87 |          END;
  18 | 1|  90 |END
  19 | 1|  90 |EOF
CODE FILE COPIED (90 WORDS)
CONSTANT TABLE COPIED (4 WORDS)
2 RECORDS WRITTEN INTO FILE 1
END OF COMPILATION FEBRUARY 12, 1975.  CLOCK TIME = 20:34:0.02.

19 CARDS WERE READ.
NO ERRORS WERE DETECTED.
```

```
***CODE SYNTHESIS FILTER***

BLOCK SELECTION METHOD WILL BE FIRST-IN-FIRST-OUT

***C O M P L E T E   T A B L E   D U M P***
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

***T A B L E   D U M P*** AT LOCATION = 0

CONSTANTS <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
CONSYM    |1   |2   |3   |4
CONTYPE   |INT |INT |INT |INT
CONVAL    |100 |101 |102 |103
CONINT    |1   |2   |3   |4

ADDR TABL <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

VALUE STACK (TOP AT 127)

CONT STK  <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
BLOCK#    |0
ENTRY     |0

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)
```

**** FORMATTED INTERMEDIATE CODE DUMP ****

LOC	OFF SET	OP CODE	CNS	ADR /ETC	RAW CODE	OPTIMIZATION TYPE
0	0	ENT	C	100	00038064	
1	0	TOGGLE		257	00080101	
2	0	LOC		1	00020001	
3	0	INT		0	00050000	
4	0	STD		0	00290000	
5	0	LOC		2	00020002	
6	0	INT		0	00050000	
7	0	STD		0	00290000	
8	0	LOC		3	00020003	
9	0	INT		1	00050001	
10	0	STO		0	00280000	
11	0	DEL		0	002A0000	
12	0	LOC		1	00020001	
13	0	LOD		0	00270000	
14	0	LOC		2	00020002	
15	0	LOD		0	00270000	
16	0	GTR		0	00210000	
17	0	XIT		73	00040049	
18	0	XIT		20	00040014	
19	0	BSC		0	002F0000	
20	0	ENT		1	00030001	
21	0	LOC		1	00020001	
22	0	LOD		0	00270000	
23	0	LOC		2	00020002	
24	0	LOD		0	00270000	
25	0	GTR		0	00210000	
26	0	XIT		44	0004002C	
27	0	XIT		29	0004001D	
28	0	BSC		0	002F0000	
29	0	ENT		1	00030001	
30	0	LOC		2	00020002	
31	0	INT		2	00050002	
32	0	STO		0	00280000	
33	0	DEL		0	002A0000	
34	0	LOC		4	00020004	
35	0	LOC		2	00020002	
36	0	LOD		0	00270000	
37	0	LOC		3	00020003	
38	0	LOD		0	00270000	
39	0	MUL		0	00140000	
40	0	STO		0	00280000	
41	0	DEL		0	002A0000	

42	0	XIT	57	00040039
43	0	BRS	0	002E0000
44	0	ENT	1	00030001
45	0	LOC	2	00020002
46	0	INT	3	00050003
47	0	STO	0	00280000
48	0	DEL	0	002A0000
49	0	LOC	8	00020008
50	0	LOC	2	00020002
51	0	LOD	0	00270000
52	0	LOC	3	00020003
53	0	LOD	0	00270000
54	0	MUL	0	00140000
55	0	STO	0	00280000
56	0	DEL	0	002A0000
57	0	ENT	1	00030001
58	0	LOC	5	00020005
59	0	LOC	2	00020002
60	0	LOD	0	00270000
61	0	STO	0	00280000
62	0	DEL	0	002A0000
63	0	LOC	6	00020006
64	0	LOC	5	00020005
65	0	LOD	0	00270000
66	0	LOC	3	00020003
67	0	LOD	0	00270000
68	0	MUL	0	00140000
69	0	STO	0	00280000
70	0	DEL	0	002A0000
71	0	XIT	86	00040056
72	0	BRS	0	002E0000
73	0	ENT	1	00030001
74	0	LOC	2	00020002
75	0	INT	4	00050004
76	0	STO	0	00280000
77	0	DEL	0	002A0000
78	0	LOC	8	00020008
79	0	LOC	2	00020002
80	0	LOD	0	00270000
81	0	LOC	3	00020003
82	0	LOD	0	00270000
83	0	MUL	0	00140000
84	0	STO	0	00280000
85	0	DEL	0	002A0000
86	0	ENT	1	00030001
87	0	LOC	7	00020007
88	0	LOC	2	00020002
89	0	LOD	0	00270000
90	0	STO	0	00280000
91	0	DEL	0	002A0000
92	0	LOC	7	00020007
93	0	LOC	7	00020007
94	0	LOD	0	00270000
95	0	LOC	3	00020003
96	0	LOD	0	00270000
97	0	MUL	0	00140000
98	0	STO	0	00280000
99	0	DEL	0	002A0000

****

BASIC BLOCK #1
BEGINNING AT 0, ENDING AT 0
BLOCK TRAVERSED 0 TIMES
0 REFERENCES:

E N D O F C O M P L E T E T A B L E D U M P

B+2	0	LOC	1	00020001
B+3	0	INT	0	00050000
B+4	0	STD	0	00290000
B+5	0	LOC	2	00020002
B+6	0	INT	0	00050000
B+7	0	STD	0	00290000
B+8	0	LOC	3	00020003
B+9	0	INT	1	00050001
B+10	0	STO	0	00280000
B+11	0	DEL	0	002A0000
B+12	0	LOC	1	00020001
B+13	0	LOD	0	00270000
B+14	0	LOC	2	00020002
B+15	0	LOD	0	00270000
B+16	0	GTR	0	00210000
B+17	0	XIT	73	00040049
B+18	0	XIT	20	00040014
B+19	0	BSC	0	002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 2 IS:

```
C(104:105,*,106)  A(3,100,100)  A(2,*,105)  A(1,*,104)

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 3 IS:
 C(104:105,*,106)  A(3,100,100)  A(2,*,105)  A(1,*,104)

 CONTROL AT 73, FROM BLK 1 TO BLK 2 (PASS 1)
C+73    0    |ENT          |C|104     |  00038068  |
B+74    0    |LOC          | |2       |  00020002  |
B+75    0    |INT          | |4       |  00050004  |
B+76    0    |STO          | |0       |  00280000  |
B+77    0    |DEL          | |0       |  002A0000  |
B+78    0    |LOC          | |8       |  00020008  |
B+79    0    |LOC          | |2       |  00020002  |
B+80    0    |LOD          | |0       |  00270000  |
B+81    0    |LOC          | |3       |  00020003  |
B+82    0    |LOD          | |0       |  00270000  |
B+83    0    |MUL          | |0       |  00140000  |
B+84    0    |STO          | |0       |  00280000  |
B+85    0    |DEL          | |0       |  002A0000  |
B+86    0    |ENT          | |1       |  00030001  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 4 IS:
 C(104:105,*,106)  C(103:100,*,103)  A(8,103,103)  A(2,103,103)  A(3,100,100)
 A(1,*,104)

 CONTROL AT 20, FROM BLK 1 TO BLK 3 (PASS 1)
C+20    0    |ENT          |C|109     |  0003806D  |
B+21    0    |LOC          | |1       |  00020001  |
B+22    0    |LOD          | |0       |  00270000  |
B+23    0    |LOC          | |2       |  00020002  |
B+24    0    |LOD          | |0       |  00270000  |
B+25    0    |GTR          | |0       |  00210000  |
B+26    0    |XIT          | |44      |  0004002C  |
B+27    0    |XIT          | |29      |  0004001D  |
B+28    0    |BSC          | |0       |  002F0000  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 5 IS:
 C(104:105,*,106)  A(3,100,100)  A(2,*,105)  A(1,*,104)

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 6 IS:
 C(104:105,*,106)  A(3,100,100)  A(2,*,105)  A(1,*,104)

 CONTROL AT 86, FROM BLK 2 TO BLK 4 (PASS 1)
C+86    0    |ENT          |C|114     |  00038072  |
B+87    0    |LOC          | |7       |  00020007  |
B+88    0    |LOC          | |2       |  00020002  |
B+89    0    |LOD          | |0       |  00270000  |
B+90    0    |STO          | |0       |  00280000  |
B+91    0    |DEL          | |0       |  002A0000  |
B+92    0    |LOC          | |7       |  00020007  |
B+93    0    |LOC          | |7       |  00020007  |
B+94    0    |LOD          | |0       |  00270000  |
B+95    0    |LOC          | |3       |  00020003  |
B+96    0    |LOD          | |0       |  00270000  |
B+97    0    |MUL          | |0       |  00140000  |
B+98    0    |STO          | |0       |  00280000  |
B+99    0    |DEL          | |0       |  002A0000  |
 CONTROL AT 44, FROM BLK 3 TO BLK 5 (PASS 1)
C+44    0    |ENT          |C|119     |  00038077  |
B+45    0    |LOC          | |2       |  00020002  |
B+46    0    |INT          | |3       |  00050003  |
B+47    0    |STO          | |0       |  00280000  |
B+48    0    |DEL          | |0       |  002A0000  |
B+49    0    |LOC          | |8       |  00020008  |
B+50    0    |LOC          | |2       |  00020002  |
B+51    0    |LOD          | |0       |  00270000  |
B+52    0    |LOC          | |3       |  00020003  |
B+53    0    |LOD          | |0       |  00270000  |
B+54    0    |MUL          | |0       |  00140000  |
B+55    0    |STO          | |0       |  00280000  |
B+56    0    |DEL          | |0       |  002A0000  |
B+57    0    |ENT          | |1       |  00030001  |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 7 IS:
 C(104:105,*,106)  C(102:100,*,102)  A(8,102,102)  A(2,102,102)  A(3,100,100)
 A(1,*,104)

 CONTROL AT 29, FROM BLK 3 TO BLK 6 (PASS 1)
C+29    0    |ENT          |C|124     |  0003807C  |
B+30    0    |LOC          | |2       |  00020002  |
B+31    0    |INT          | |2       |  00050002  |
B+32    0    |STO          | |0       |  00280000  |
B+33    0    |DEL          | |0       |  002A0000  |
B+34    0    |LOC          | |4       |  00020004  |
B+35    0    |LOC          | |2       |  00020002  |
B+36    0    |LOD          | |0       |  00270000  |
```

```
B+37    0   |LOC         | |3         | 00020003 |
B+38    0   |LOD         | |0         | 00270000 |
B+39    0   |MUL         | |0         | 00140000 |
B+40    0   |STO         | |0         | 00280000 |
B+41    0   |DEL         | |0         | 002A0000 |
B+42    0   |XIT         | |57        | 00040039 |
B+43    0   |BRS         | |0         | 002E0000 |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 6 IS:
 C(104:105,*,106)  C(101:100,*,101)  A(4,101,101)  A(2,101,101)  A(3,100,100)
 A(1,*,104)

 CURRENT POOL FOR BLOCK# 7 IS:
 C(104:105,*,106)  C(102:100,*,102)  A(8,102,102)  A(2,102,102)  A(3,100,100)
 A(1,*,104)

AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 7 IS:
 A(2,*,110)  A(3,100,100)  A(1,*,104)  C(110:100,*,110)


 CONTROL AT 57, FROM BLK 6 TO BLK 7 (PASS 1)
C+57    0   |ENT         |C|129       | 00038081 |
B+58    0   |LOC         | |5         | 00020005 |
B+59    0   |LOC         | |2         | 00020002 |
B+60    0   |LOD         | |0         | 00270000 |
B+61    0   |STO         | |0         | 00280000 |
B+62    0   |DEL         | |0         | 002A0000 |
B+63    0   |LOC         | |6         | 00020006 |
B+64    0   |LOC         | |5         | 00020005 |
B+65    0   |LOD         | |0         | 00270000 |
B+66    0   |LOC         | |3         | 00020003 |
B+67    0   |LOD         | |0         | 00270000 |
B+68    0   |MUL         | |0         | 00140000 |
B+69    0   |STO         | |0         | 00280000 |
B+70    0   |DEL         | |0         | 002A0000 |
B+71    0   |XIT         | |86        | 00040056 |
B+72    0   |BRS         | |0         | 002E0000 |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 7 IS:
 C(110:100,*,110)  A(6,*,110)  A(5,*,110)  A(2,*,110)  A(3,100,100)
 A(1,*,104)

 CURRENT POOL FOR BLOCK# 4 IS:
 C(104:105,*,106)  C(103:100,*,103)  A(8,103,103)  A(2,103,103)  A(3,100,100)
 A(1,*,104)

AFTER PERFORMING THE MEET OPERATION:

 NEW CURRENT POOL/INPUT POOL FOR  BLOCK# 4 IS:
 A(2,*,111)  A(3,100,100)  A(1,*,104)  C(111:100,*,111)


 CONTROL AT 86, FROM BLK 7 TO BLK 4 (PASS 2)
C+86    0   |ENT         |C|114       | 00038072 |
B+87    0   |LOC         | |7         | 00020007 |
B+88    0   |LOC         | |2         | 00020002 |
B+89    0   |LOD         | |0         | 00270000 |
B+90    0   |STO         | |0         | 00280000 |
B+91    0   |DEL         | |0         | 002A0000 |
B+92    0   |LOC         | |7         | 00020007 |
B+93    0   |LOC         | |7         | 00020007 |
B+94    0   |LOD         | |0         | 00270000 |
B+95    0   |LOC         | |3         | 00020003 |
B+96    0   |LOD         | |0         | 00270000 |
B+97    0   |MUL         |C|1000      | 001483E8 |   COMM SUBEXP ELIM & CONS PROP
B+98    0   |STO         | |0         | 00280000 |
B+99    0   |DEL         | |0         | 002A0000 |

                         FINAL OPTIMIZATION RESULTS
#############################################################################
-----------------------------------------------------------------------------

OPTIMIZATION AT 25
          (104,105|16,106)     COMM SUBEXP ELIM

OPTIMIZATION AT 39
          (100,101|39,101)     CONSTANT PROPAGATION
```

```
OPTIMIZATION AT 54
        (102,100|54,102)     CONSTANT PROPAGATION

OPTIMIZATION AT 68
        (110,100|68,110)     SIMPLIFICATION

OPTIMIZATION AT 83
        (100,103|83,103)     CONSTANT PROPAGATION

OPTIMIZATION AT 97
        (100,111|97,111)     SIMPLIFICATION
```

**** FORMATTED INTERMEDIATE CODE DUMP ****

LOC	OFF SET	OP CODE	CNS	ADR /ETC	RAW CODE	OPTIMIZATION TYPE
0	0	ENT		100	00030064	
1	0	TOGGLE		257	00080101	
2	0	LOC		1	00020001	
3	0	INT		0	00050000	
4	0	STD		0	00290000	
5	0	LOC		2	00020002	
6	0	INT		0	00050000	
7	0	STD		0	00290000	
8	0	LOC		3	00020003	
9	0	INT		1	00050001	
10	0	STO		0	00280000	
11	0	DEL		0	002A0000	
12	0	LOC		1	00020001	
13	0	LOD		0	00270000	
14	0	LOC		2	00020002	
15	0	LOD		0	00270000	
16	0	GTR		1024	00210400	
17	0	XIT		73	00040049	
18	0	XIT		20	00040014	
19	0	BSC		0	002F0000	
20	0	ENT		109	0003006D	
21	0	LOC		1	00020001	
22	0	LOD		0	00270000	
23	0	LOC		2	00020002	
24	0	LOD		0	00270000	
25	0	GTR	C	1008	002183F0	COMM SUBEXP ELIM
26	0	XIT		44	0004002C	
27	0	XIT		29	0004001D	
28	0	BSC		0	002F0000	
29	0	ENT		124	0003007C	
30	0	LOC		2	00020002	
31	0	INT		2	00050002	
32	0	STO		0	00280000	
33	0	DEL		0	002A0000	
34	0	LOC		4	00020004	
35	0	LOC		2	00020002	
36	0	LOD		0	00270000	
37	0	LOC		3	00020003	
38	0	LOD		0	00270000	
39	0	MUL	C	988	001483DC	CONSTANT PROPAGATION
40	0	STO		0	00280000	
41	0	DEL		0	002A0000	
42	0	XIT		57	00040039	
43	0	BRS		0	002E0000	
44	0	ENT		119	00030077	
45	0	LOC		2	00020002	
46	0	INT		3	00050003	
47	0	STO		0	00280000	
48	0	DEL		0	002A0000	
49	0	LOC		8	00020008	
50	0	LOC		2	00020002	
51	0	LOD		0	00270000	
52	0	LOC		3	00020003	
53	0	LOD		0	00270000	
54	0	MUL	C	996	001483E4	CONSTANT PROPAGATION
55	0	STO		0	00280000	
56	0	DEL		0	002A0000	
57	0	ENT		129	00030081	
58	0	LOC		5	00020005	
59	0	LOC		2	00020002	
60	0	LOD		0	00270000	
61	0	STO		0	00280000	
62	0	DEL		0	002A0000	
63	0	LOC		6	00020006	
64	0	LOC		5	00020005	
65	0	LOD		0	00270000	
66	0	LOC		3	00020003	
67	0	LOD		0	00270000	
68	0	MUL	C	984	001483D8	SIMPLIFICATION
69	0	STO		0	00280000	
70	0	DEL		0	002A0000	

```
71   0  |[illegible]     |  |86    | 00040056 |
72   0  |BRS             |  |0     | 002E0000 |
73   0  |ENT             |  |104   | 00030068 |
74   0  |LOC             |  |2     | 00020002 |
75   0  |INT             |  |4     | 00050004 |
76   0  |STO             |  |0     | 00280000 |
77   0  |DEL             |  |0     | 002A0000 |
78   0  |LOC             |  |8     | 00020008 |
79   0  |LOC             |  |2     | 00020002 |
80   0  |LOD             |  |0     | 00270000 |
81   0  |LOC             |  |3     | 00020003 |
82   0  |LOD             |  |0     | 00270000 |
83   0  |MUL             |C |1016  | 001483F8 |   CONSTANT PROPAGATION
84   0  |STO             |  |0     | 00280000 |
85   0  |DEL             |  |0     | 002A0000 |
86   0  |ENT             |  |114   | 00030072 |
87   0  |LOC             |  |7     | 00020007 |
88   0  |LOC             |  |2     | 00020002 |
89   0  |LOD             |  |0     | 00270000 |
90   0  |STO             |  |0     | 00280000 |
91   0  |DEL             |  |0     | 002A0000 |
92   0  |LOC             |  |7     | 00020007 |
93   0  |LOC             |  |7     | 00020007 |
94   0  |LOD             |  |0     | 00270000 |
95   0  |LOC             |  |3     | 00020003 |
96   0  |LOD             |  |0     | 00270000 |
97   0  |MUL             |C |1000  | 001483E8 |   SIMPLIFICATION
98   0  |STO             |  |0     | 00280000 |
99   0  |DEL             |  |0     | 002A0000 |
************************************************************************************
```

```
      FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

 FINAL POOL FOR  BLOCK# 1 IS:
(NULL)


 FINAL POOL FOR  BLOCK# 2 IS:
 C(104:105,*,106)  A(3,100,100)  A(2,*,105)  A(1,*,104)


 FINAL POOL FOR  BLOCK# 3 IS:
 C(104:105,*,106)  A(3,100,100)  A(2,*,105)  A(1,*,104)


 FINAL POOL FOR  BLOCK# 4 IS:
 A(2,*,111)  A(3,100,100)  A(1,*,104)  C(111:100,*,111)


 FINAL POOL FOR  BLOCK# 5 IS:
 C(104:105,*,106)  A(3,100,100)  A(2,*,105)  A(1,*,104)


 FINAL POOL FOR  BLOCK# 6 IS:
 C(104:105,*,106)  A(3,100,100)  A(2,*,105)  A(1,*,104)


 FINAL POOL FOR  BLOCK# 7 IS:
 A(2,*,110)  A(3,100,100)  A(1,*,104)  C(110:100,*,110)


     BASIC BLOCK #1
           BEGINNING AT 0, ENDING AT 19
           BLOCK TRAVERSED 1 TIMES
           0 REFERENCES:
     BASIC BLOCK #2
           BEGINNING AT 73, ENDING AT 86
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
             1
     BASIC BLOCK #3
           BEGINNING AT 20, ENDING AT 28
           BLOCK TRAVERSED 1 TIMES
           1 REFERENCES:
             1
     BASIC BLOCK #4
```

```
        BEGINNING AT 86, ENDING AT 99
        BLOCK TRAVERSED 2 TIMES
        1 REFERENCES:
            2
   BASIC BLOCK #5
        BEGINNING AT 44, ENDING AT 57
        BLOCK TRAVERSED 1 TIMES
        1 REFERENCES:
            3
   BASIC BLOCK #6
        BEGINNING AT 29, ENDING AT 43
        BLOCK TRAVERSED 1 TIMES
        1 REFERENCES:
            3
   BASIC BLOCK #7
        BEGINNING AT 57, ENDING AT 72
        BLOCK TRAVERSED 1 TIMES
        1 REFERENCES:
            6

BLOCK SUMMARY DATA:

BLOCK NUMBER                     NUMBER OF PASSES

    1                                  1
    2                                  1
    3                                  1
    4                                  2
    5                                  1
    6                                  1
    7                                  1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 8
USING THE FIRST-IN-FIRST-OUT  BLOCK SELECTION ALGORITHM.

TIME FOR THE OPTIMIZATION WAS 0.449 SECONDS.

                    ***CODE SYNTHESIS FILTER IS COMPLETE***
```

```
CARD |BL| SYL |
   1 | 0|   0 |COMMENT     TEST PROGRAM # 4         ;
$EXECUTE
   3 | 0|   1 |BEGIN LOCAL A,B,C,D,E,F,G,H,I,J,K;
   4 | 1|   1 |READ(A, B, C);
   5 | 1|  10 |I := B * C;
   6 | 1|  18 |WHILE A GTR B DO
   7 | 1|  28 |BEGIN
   8 | 1|  28 |    WHILE A GTR B DO
   9 | 2|  38 |    BEGIN
  10 | 2|  38 |        WHILE A GTR B DO
  11 | 3|  48 |            J := B * C;
  12 | 3|  59 |        K := C * B;
  13 | 3|  67 |    END;
  14 | 2|  70 |END;
  15 | 1|  73 |END
  16 | 1|  73 |EOF
CODE FILE COPIED (73 WORDS)
CONSTANT TABLE COPIED (0 WORDS)
2 RECORDS WRITTEN INTO FILE 1
END OF COMPILATION FEBRUARY 12, 1975.  CLOCK TIME = 21:13:33.31.

16 CARDS WERE READ.
NO ERRORS WERE DETECTED.
```

```
                    ***CODE SYNTHESIS FILTER***

BLOCK SELECTION METHOD WILL BE MAXIMUM CURRENT POOL

***C O M P L E T E   T A B L E   D U M P***
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

***T A B L E   D U M P*** AT LOCATION = 0

CONSTANTS <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

ADDR TABL <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)

VALUE STACK (TOP AT 127)

CONT STK  <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
BLOCK#    |0
ENTRY     |0

EXEC STAC <1>  <2>  <3>  <4>  <5>  <6>  <7>  <8>  <9>  <10> <11> <12> <13> <14>
(EMPTY)


 ****  FORMATTED INTERMEDIATE CODE DUMP  ****
 LOC OFF     OP CODE     CNS ADR     RAW CODE      OPTIMIZATION
     SET                     /ETC                       TYPE
*************************************************************************
 0      0    |ENT        |C|74       | 0003804A  |
 1      0    |TOGGLE     | |257      | 000B0101  |
 2      0    |LOC        | |1        | 00020001  |
 3      0    |INT        | |0        | 00050000  |
 4      0    |STD        | |0        | 00290000  |
 5      0    |LOC        | |2        | 00020002  |
 6      0    |INT        | |0        | 00050000  |
 7      0    |STD        | |0        | 00290000  |
 8      0    |LOC        | |3        | 00020003  |
 9      0    |INT        | |0        | 00050000  |
 10     0    |STD        | |0        | 00290000  |
 11     0    |LOC        | |9        | 00020009  |
 12     0    |LOC        | |2        | 00020002  |
 13     0    |LOD        | |0        | 00270000  |
 14     0    |LOC        | |3        | 00020003  |
 15     0    |LOD        | |0        | 00270000  |
 16     0    |MUL        | |0        | 00140000  |
 17     0    |STO        | |0        | 00280000  |
 18     0    |DEL        | |0        | 002A0000  |
 19     0    |ENT        | |1        | 00030001  |
 20     0    |LOC        | |1        | 00020001  |
 21     0    |LOD        | |0        | 00270000  |
 22     0    |LOC        | |2        | 00020002  |
 23     0    |LOD        | |0        | 00270000  |
 24     0    |GTR        | |0        | 00210000  |
 25     0    |XIT        | |73       | 00040049  |
 26     0    |XIT        | |28       | 0004001C  |
 27     0    |BSC        | |0        | 002F0000  |
 28     0    |ENT        | |1        | 00030001  |
 29     0    |ENT        | |1        | 00030001  |
 30     0    |LOC        | |1        | 00020001  |
 31     0    |LOD        | |0        | 00270000  |
 32     0    |LOC        | |2        | 00020002  |
 33     0    |LOD        | |0        | 00270000  |
 34     0    |GTR        | |0        | 00210000  |
 35     0    |XIT        | |70       | 00040046  |
 36     0    |XIT        | |38       | 00040026  |
 37     0    |BSC        | |0        | 002F0000  |
 38     0    |ENT        | |1        | 00030001  |
 39     0    |ENT        | |1        | 00030001  |
 40     0    |LOC        | |1        | 00020001  |
 41     0    |LOD        | |0        | 00270000  |
 42     0    |LOC        | |2        | 00020002  |
 43     0    |LOD        | |0        | 00270000  |
 44     0    |GTR        | |0        | 00210000  |
```

```
45    0   |XIT          | |59     | 0004003B  |
46    0   |XIT          | |48     | 00040030  |
47    0   |BSC          | |0      | 002F0000  |
48    0   |ENT          | |1      | 00030001  |
49    0   |LOC          | |10     | 0002000A  |
50    0   |LOC          | |2      | 00020002  |
51    0   |LOD          | |0      | 00270000  |
52    0   |LOC          | |3      | 00020003  |
53    0   |LOD          | |0      | 00270000  |
54    0   |MUL          | |0      | 00140000  |
55    0   |STO          | |0      | 00280000  |
56    0   |DEL          | |0      | 002A0000  |
57    0   |XIT          | |39     | 00040027  |
58    0   |BRS          | |0      | 002E0000  |
59    0   |ENT          | |1      | 00030001  |
60    0   |LOC          | |11     | 0002000B  |
61    0   |LOC          | |3      | 00020003  |
62    0   |LOD          | |0      | 00270000  |
63    0   |LOC          | |2      | 00020002  |
64    0   |LOD          | |0      | 00270000  |
65    0   |MUL          | |0      | 00140000  |
66    0   |STO          | |0      | 00280000  |
67    0   |DEL          | |0      | 002A0000  |
68    0   |XIT          | |29     | 0004001D  |
69    0   |BRS          | |0      | 002E0000  |
70    0   |ENT          | |1      | 00030001  |
71    0   |XIT          | |19     | 00040013  |
72    0   |BRS          | |0      | 002E0000  |
73    0   |ENT          | |1      | 00030001  |
*******************************************************************************

      BASIC BLOCK #1
            BEGINNING AT 0, ENDING AT 0
            BLOCK TRAVERSED 0 TIMES
            0 REFERENCES:

***E N D   O F   C O M P L E T E   T A B L E   D U M P***

B+2   0   |LOC          | |1      | 00020001  |
B+3   0   |INT          | |0      | 00050000  |
B+4   0   |STO          | |0      | 00290000  |
B+5   0   |LOC          | |2      | 00020002  |
B+6   0   |INT          | |0      | 00050000  |
B+7   0   |STO          | |0      | 00290000  |
B+8   0   |LOC          | |3      | 00020003  |
B+9   0   |INT          | |0      | 00050000  |
B+10  0   |STO          | |0      | 00290000  |
B+11  0   |LOC          | |9      | 00020009  |
B+12  0   |LOC          | |2      | 00020002  |
B+13  0   |LOD          | |0      | 00270000  |
B+14  0   |LOC          | |3      | 00020003  |
B+15  0   |LOD          | |0      | 00270000  |
B+16  0   |MUL          | |0      | 00140000  |
B+17  0   |STO          | |0      | 00280000  |
B+18  0   |DEL          | |0      | 002A0000  |
B+19  0   |ENT          | |1      | 00030001  |

  INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 2 IS:
  C(75:76,*,77)  A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)

  CONTROL AT 19, FROM BLK 1 TO BLK 2 (PASS 1)
C+19  0   |ENT          |C|78     | 0003804E  |
B+20  0   |LOC          | |1      | 00020001  |
B+21  0   |LOD          | |0      | 00270000  |
B+22  0   |LOC          | |2      | 00020002  |
B+23  0   |LOD          | |0      | 00270000  |
B+24  0   |GTR          | |0      | 00210000  |
B+25  0   |XIT          | |73     | 00040049  |
B+26  0   |XIT          | |28     | 0004001C  |
B+27  0   |BSC          | |0      | 002F0000  |

  INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 3 IS:
  C(75:76,*,77)  C(74:75,*,78)  A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)

  INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 4 IS:
  C(75:76,*,77)  C(74:75,*,78)  A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)

  CONTROL AT 28, FROM BLK 2 TO BLK 4 (PASS 1)
C+28  0   |ENT          |C|88     | 00038058  |
B+29  0   |ENT          | |1      | 00030001  |

  INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 5 IS:
  C(74:75,*,78)  C(75:76,*,77)  A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)

  CONTROL AT 29, FROM BLK 4 TO BLK 5 (PASS 1)
C+29  0   |ENT          |C|93     | 0003805D  |
```

```
B+30   0   |LOC          | |1      | 00020001 |
B+31   0   |LOD          | |0      | 00270000 |
B+32   0   |LOC          | |2      | 00020002 |
B+33   0   |LOD          | |0      | 00270000 |
B+34   0   |GTR          | |0      | 00210000 |
B+35   0   |XIT          | |70     | 00040046 |
B+36   0   |XIT          | |38     | 00040026 |
B+37   0   |BSC          | |0      | 002F0000 |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 6 IS:
 C(75:76,*,77)  C(74:75,*,78)  A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 7 IS:
 C(75:76,*,77)  C(74:75,*,78)  A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)

 CONTROL AT 38, FROM BLK 5 TO BLK 7 (PASS 1)
C+38   0   |ENT          |C|103    | 00038067 |
B+39   0   |ENT          | |1      | 00030001 |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 8 IS:
 C(74:75,*,78)  C(75:76,*,77)  A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)

 CONTROL AT 39, FROM BLK 7 TO BLK 8 (PASS 1)
C+39   0   |ENT          |C|108    | 0003806C |
B+40   0   |LOC          | |1      | 00020001 |
B+41   0   |LOD          | |0      | 00270000 |
B+42   0   |LOC          | |2      | 00020002 |
B+43   0   |LOD          | |0      | 00270000 |
B+44   0   |GTR          | |0      | 00210000 |
B+45   0   |XIT          | |59     | 0004003B |
B+46   0   |XIT          | |48     | 00040030 |
B+47   0   |BSC          | |0      | 002F0000 |

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 9 IS:
 C(75:76,*,77)  C(74:75,*,78)  A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)

 INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR  BLOCK# 10 IS:
 C(75:76,*,77)  C(74:75,*,78)  A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)

 CONTROL AT 48, FROM BLK 8 TO BLK 10 (PASS 1)
C+48   0   |ENT          |C|118    | 00038076 |
B+49   0   |LOC          | |10     | 0002000A |
B+50   0   |LOC          | |2      | 00020002 |
B+51   0   |LOD          | |0      | 00270000 |
B+52   0   |LOC          | |3      | 00020003 |
B+53   0   |LOD          | |0      | 00270000 |
B+54   0   |MUL          | |0      | 00140000 |
B+55   0   |STO          | |0      | 00280000 |
B+56   0   |DEL          | |0      | 002A0000 |
B+57   0   |XIT          | |39     | 00040027 |
B+58   0   |BRS          | |0      | 002E0000 |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 10 IS:
 C(74:75,*,78)  C(75:76,*,77)  A(10,*,77)  A(9,*,77)  A(3,*,76)  A(2,*,75)
 A(1,*,74)

 CURRENT POOL FOR BLOCK# 8 IS:
 C(74:75,*,78)  C(75:76,*,77)  A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)
AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

 CONTROL AT 59, FROM BLK 8 TO BLK 9 (PASS 1)
C+59   0   |ENT          |C|113    | 00038071 |
B+60   0   |LOC          | |11     | 0002000B |
B+61   0   |LOC          | |3      | 00020003 |
B+62   0   |LOD          | |0      | 00270000 |
B+63   0   |LOC          | |2      | 00020002 |
B+64   0   |LOD          | |0      | 00270000 |
B+65   0   |MUL          | |0      | 00140000 |
B+66   0   |STO          | |0      | 00280000 |
B+67   0   |DEL          | |0      | 002A0000 |
B+68   0   |XIT          | |29     | 0004001D |
B+69   0   |BRS          | |0      | 002E0000 |
BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 9 IS:
 C(74:75,*,78)  C(75:76,*,77)  A(11,*,77)  A(9,*,77)  A(3,*,76)  A(2,*,75)
 A(1,*,74)

 CURRENT POOL FOR BLOCK# 5 IS:
```

```
 C(74:75,*,78)  C(75:76,*,77)  A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)

AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

 CONTROL AT 70, FROM BLK 5 TO BLK 6 (PASS 1)
C+70    0    |ENT          |C|98      | 00038062  |
B+71    0    |XIT          | |19      | 00040013  |
B+72    0    |BRS          | |0       | 002E0000  |

BEFORE THE MEET OPERATION:

 OUTPUT POOL FROM BLOCK# 6 IS:
 C(74:75,*,78)  C(75:76,*,77)  A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)

 CURRENT POOL FOR BLOCK# 2 IS:
 C(75:76,*,77)  A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)

AFTER PERFORMING THE MEET OPERATION:
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

 CONTROL AT 73, FROM BLK 2 TO BLK 3 (PASS 1)
C+73    0    |ENT          |C|83      | 00038053  |

                                    FINAL OPTIMIZATION RESULTS
################################################################################
--------------------------------------------------------------------------------

OPTIMIZATION AT 34
          (74,75|24,78)         COMM SUBEXP ELIM

OPTIMIZATION AT 44
          (74,75|24,78)         COMM SUBEXP ELIM

OPTIMIZATION AT 54
          (75,76|16,77)         COMM SUBEXP ELIM

OPTIMIZATION AT 65
          (75,76|16,77)         COMM SUBEXP ELIM

 ****  FORMATTED INTERMEDIATE CODE DUMP   ****
 LOC OFF     OP CODE      CNS ADR     RAW CODE       OPTIMIZATION
     SET                      /ETC                       TYPE
**************************************************************************
 0      0    |ENT          | |74      | 0003004A  |
 1      0    |TOGGLE       | |257     | 000B0101  |
 2      0    |LOC          | |1       | 00020001  |
 3      0    |INT          | |0       | 00050000  |
 4      0    |STD          | |0       | 00290000  |
 5      0    |LOC          | |2       | 00020002  |
 6      0    |INT          | |0       | 00050000  |
 7      0    |STD          | |0       | 00290000  |
 8      0    |LOC          | |3       | 00020003  |
 9      0    |INT          | |0       | 00050000  |
 10     0    |STD          | |0       | 00290000  |
 11     0    |LOC          | |9       | 00020009  |
 12     0    |LOC          | |2       | 00020002  |
 13     0    |LOD          | |0       | 00270000  |
 14     0    |LOC          | |3       | 00020003  |
 15     0    |LOD          | |0       | 00270000  |
 16     0    |MUL          | |1024    | 00140400  |
 17     0    |STO          | |0       | 00280000  |
 18     0    |DEL          | |0       | 002A0000  |
 19     0    |ENT          | |78      | 0003004E  |
 20     0    |LOC          | |1       | 00020001  |
 21     0    |LOD          | |0       | 00270000  |
 22     0    |LOC          | |2       | 00020002  |
 23     0    |LOD          | |0       | 00270000  |
 24     0    |GTR          | |1018    | 002103FA  |
 25     0    |XIT          | |73      | 00040049  |
 26     0    |XIT          | |28      | 0004001C  |
 27     0    |BSC          | |0       | 002F0000  |
 28     0    |ENT          | |88      | 00030058  |
 29     0    |ENT          | |93      | 0003005D  |
 30     0    |LOC          | |1       | 00020001  |
 31     0    |LOD          | |0       | 00270000  |
 32     0    |LOC          | |2       | 00020002  |
 33     0    |LOD          | |0       | 00270000  |
 34     0    |GTR          |C|1002    | 002183EA  |    COMM SUBEXP ELIM
 35     0    |XIT          | |70      | 00040046  |
 36     0    |XIT          | |38      | 00040026  |
 37     0    |BSC          | |0       | 002F0000  |
 38     0    |ENT          | |103     | 00030067  |
 39     0    |ENT          | |108     | 0003006C  |
```

40	0	LOC		1	00020001	
41	0	LOD		0	00270000	
42	0	LOC		2	00020002	
43	0	LOD		0	00270000	
44	0	GTR	C	986	002183DA	COMM SUBEXP ELIM
45	0	XIT		59	0004003B	
46	0	XIT		48	00040030	
47	0	BSC		0	002F0000	
48	0	ENT		118	00030076	
49	0	LOC		10	0002000A	
50	0	LOC		2	00020002	
51	0	LOD		0	00270000	
52	0	LOC		3	00020003	
53	0	LOD		0	00270000	
54	0	MUL	C	974	001483CE	COMM SUBEXP ELIM
55	0	STO		0	00280000	
56	0	DEL		0	002A0000	
57	0	XIT		39	00040027	
58	0	BRS		0	002E0000	
59	0	ENT		113	00030071	
60	0	LOC		11	0002000B	
61	0	LOC		3	00020003	
62	0	LOD		0	00270000	
63	0	LOC		2	00020002	
64	0	LOD		0	00270000	
65	0	MUL	C	970	001483CA	COMM SUBEXP ELIM
66	0	STO		0	00280000	
67	0	DEL		0	002A0000	
68	0	XIT		29	0004001D	
69	0	BRS		0	002E0000	
70	0	ENT		98	00030062	
71	0	XIT		19	00040013	
72	0	BRS		0	002E0000	
73	0	ENT		83	00030053	

```
*****************************************************************************************

        FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

 FINAL POCL FOR  BLOCK# 1 IS:
(NULL)


 FINAL POCL FOR  BLOCK# 2 IS:
 A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)  C(75:76,*,77)


 FINAL POCL FOR  BLOCK# 3 IS:
 C(75:76,*,77)  C(74:75,*,78)  A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)


 FINAL POCL FOR  BLOCK# 4 IS:
 C(75:76,*,77)  C(74:75,*,78)  A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)


 FINAL POCL FOR  BLOCK# 5 IS:
 A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)  C(74:75,*,78)  C(75:76,*,77)


 FINAL POCL FOR  BLOCK# 6 IS:
 C(75:76,*,77)  C(74:75,*,78)  A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)


 FINAL POCL FOR  BLOCK# 7 IS:
 C(75:76,*,77)  C(74:75,*,78)  A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)


 FINAL POCL FOR  BLOCK# 8 IS:
 A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)  C(74:75,*,78)  C(75:76,*,77)


 FINAL POCL FOR  BLOCK# 9 IS:
 C(75:76,*,77)  C(74:75,*,78)  A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)
```

```
FINAL POOL FOR  BLOCK# 10 IS:
C(75:76,*,77)  C(74:75,*,78)  A(9,*,77)  A(3,*,76)  A(2,*,75)  A(1,*,74)


      BASIC BLOCK #1
            BEGINNING AT 0, ENDING AT 19
            BLOCK TRAVERSED 1 TIMES
            0 REFERENCES:
      BASIC BLOCK #2
            BEGINNING AT 19, ENDING AT 27
            BLOCK TRAVERSED 1 TIMES
            1 REFERENCES:
                1
      BASIC BLOCK #3
            BEGINNING AT 73, ENDING AT 73
            BLOCK TRAVERSED 1 TIMES
            1 REFERENCES:
                2
      BASIC BLOCK #4
            BEGINNING AT 28, ENDING AT 29
            BLOCK TRAVERSED 1 TIMES
            1 REFERENCES:
                2
      BASIC BLOCK #5
            BEGINNING AT 29, ENDING AT 37
            BLOCK TRAVERSED 1 TIMES
            1 REFERENCES:
                4
      BASIC BLOCK #6
            BEGINNING AT 70, ENDING AT 72
            BLOCK TRAVERSED 1 TIMES
            1 REFERENCES:
                5
      BASIC BLOCK #7
            BEGINNING AT 38, ENDING AT 39
            BLOCK TRAVERSED 1 TIMES
            1 REFERENCES:
                5
      BASIC BLOCK #8
            BEGINNING AT 39, ENDING AT 47
            BLOCK TRAVERSED 1 TIMES
            1 REFERENCES:
                7
      BASIC BLOCK #9
            BEGINNING AT 59, ENDING AT 69
            BLOCK TRAVERSED 1 TIMES
            1 REFERENCES:
                8
      BASIC BLOCK #10
            BEGINNING AT 48, ENDING AT 58
            BLOCK TRAVERSED 1 TIMES
            1 REFERENCES:
                8

BLOCK SUMMARY DATA:

BLOCK NUMBER                        NUMBER OF PASSES

     1                                     1
     2                                     1
     3                                     1
     4                                     1
     5                                     1
     6                                     1
     7                                     1
     8                                     1
     9                                     1
     10                                    1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 10
USING THE MAXIMUM CURRENT POOL  BLOCK SELECTION ALGORITHM.

TIME FOR THE OPTIMIZATION WAS 0.185 SECONDS.

                    ***CODE SYNTHESIS FILTER IS COMPLETE***
```

```
                /*<><><><><><><><><><><><><><><><><><><><><>*/
                /*                                          */
                /* C O D E   S Y N T H E S I S   F I L T E R */
                /*                                          */
                /*   THIS PROGRAM OPERATES ON AN INTERMEDIATE*/
                /*   LANGUAGE RECEIVED FROM THE ALGOL-E      */
                /*   COMPILER IN THE FOLLOWING FORM          */
                /*                                          */
                /*   BITS      8       8      .1.      15    */
                /*         OFFSET  OPCODE  C-FIELD  ADDRESS  */
                /*                 / TYPE                    */
                /*   USING THE META - EXECUTION TECHNIQUE,   */
                /*   THE ALGORITHM (Q) OF KILDALL IS         */
                /*   APPLIED TO THE INTERMEDIATE LANGUAGE.   */
                /*   NO CODE OPTIMIZATION IS IN FACT DONE;   */
                /*   RATHER UPON RECOGNITION OF A POSSIBLE   */
                /*   CODE INEFFEICIENCY BIT 16 IN THE 32 BIT */
                /*   INTERMEDIATE LANGUAGE IS SET TO INDICATE*/
                /*   THAT DURING CODE GENERATION AN          */
                /*   OPTIMIZATION CAN BE ACCOMPLISHED.       */
                /*   IMPORTANT DATA DESCRIPTOR DEFINITIONS:  */
                /*       EXEC     = EXECUTION STACK FOR THE  */
                /*                  META EXECUTION           */
                /*       CSFCODE = BUFFER FOR ALGOL-E CODE   */
                /*       ADDRESS = VARIABLE DEFINITION TABLES */
                /*       CON      = CONSTANT TABLES          */
                /*       CONTROL = THE LIST OF BLOCKS TO BE  */
                /*                  PROCESSED...PGM TERMINATES */
                /*                  WHEN THIS TABLE IS PAU   */
                /*       STATE    = POINTER TABLE TO CURRENT */
                /*                  STATE INFO FOR EACH BLOCK */
                /*                  PROCESSED                */
                /*       VTAB     = EXPRESSION STACK FOR ALL */
                /*                  EXPRESSIONS              */
                /*       EX       = EXECUTION STACK FOR THE  */
                /*                  META-EXECUTIONS          */
                /*   OPTIM_TYPE = CODE FOR THE OPTIMIZATION  */
                /*                  TYPE DETECTED            */
                /*                                          */
                /*<><><><><><><><><><><><><><><><><><><><><>*/
                /*                                          */
                /* THE FOLLOWING TABLES ARE OUTPUT FROM THE */
                /* PROGRAM "GENERATOR" AND ARE USED TO      */
                /* DESCRIBE THE INTERMEDIATE LANGUAGE.      */
                /*                                          */
                /*<><><><><><><><><><><><><><><><><><><><><>*/
                /*                                          */
                /* BASIC OPERATORS SET TO 0,1,ETC IN INITIAL*/
                /*                                          */
                /*<><><><><><><><><><><><><><><><><><><><><>*/
                /*                                          */
DECLARE(NULL, DEFAULT, LOC, ENT, XIT, INT, REAL, BOOL, IARRAY, RARRAY) BIT(8),
         NTYPES LITERALLY'9',
                /*                                          */
                /*<><><><><><><><><><><><><><><><><><><><><>*/
                /*                                          */
                /*   REMAINDER OF THE OPERATIONS CODES      */
                /*                                          */
                /*<><><><><><><><><><><><><><><><><><><><><>*/
                /*                                          */
         (REFER, TOGGLE, PASS, TRU, RND, FLT, ADD, RADD, SUB, RSUB, MUL, RMUL, DIV
, RDIV, EXP, RIXP, IRXP, RRXP, LSS, LEQ, EQL, NEQ, GEQ, GTR, INEG, RNEG, NOT,
AND, BOR, LOD, STO, STD, DEL, DUP, XCH, HLT, BRS, BSC, NOP, PRO, RTN, GET, RET,
RRD, IRD, BRD, WRV, DMP, TAB, SUP)BIT(8),
         NOPCODES LITERALLY'50',
                /*                                          */
                /*<><><><><><><><><><><><><><><><><><><><><>*/
                /*                                          */
                /* CHARACTER STRINGS OF LEGAL OPCODES -     */
                /* OPRANGE IS USED TO ACCESS                */
                /*                                          */
                /*<><><><><><><><><><><><><><><><><><><><><>*/
                /*                                          */
         OPCODES CHARACTER INITIAL(
'NULLDEFAULTLOCENTXITINTREALBOOLIARRAYRARRAYREFERTOGGLEPASSTRURNDFLTADDRADDSUBRS
UBMULRMULDIVRDIVEXPRIXPIRXPRRXPLSSLEQEQLNEQGEQGTRINEGRNEGNOTANDBORLODSTOSTDDELDU
PXCHHLTBRSBSCNOPPRORTNGETRETRRDIRDBRDWRVDMPTABSUP'),
```

```
                    /*                                          */
                    /*<><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                          */
                    /* START CHAR POSIT OF OP CODES IN TABLE    */
                    /* OPCODES:                                 */
                    /*         I = FIRST CHARACTER              */
                    /*     I + 1 = LAST CHARACTER + 1           */
                    /*                                          */
                    /*<><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                          */
          OPRANGE(60) BIT(8) INITIAL(0, 4, 11, 14, 17, 20, 23, 27, 31, 37, 43, 48
, 54, 58, 61, 64, 67, 70, 74, 77, 81, 84, 88, 91, 95, 98, 102, 106, 110, 113,
116, 119, 122, 125, 128, 132, 136, 139, 142, 145, 148, 151, 154, 157, 160, 163,
166, 169, 172, 175, 178, 181, 184, 187, 190, 193, 196, 199, 202, 205, 208),
                    /*                                          */
                    /*<><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                          */
                    /* TYPE OF OPERATION FOR CASE STATEMENTS,   */
                    /* ETC.                                     */
                    /*                                          */
                    /*<><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                          */
          OP_TYPE(59) BIT(8) INITIAL(0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 7, 7,
7, 8, 8, 0, 0, 8, 8, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 8, 8, 2, 4, 3
, 5, 6, 9, 0, 11, 10, 0, 12, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                    /*                                          */
                    /*<><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                          */
                    /*              DEGREE OF OPCODE            */
                    /* OPERATORS NECESSARY TO PERFORM A         */
                    /* PARTICULAR OP CODE.  STORED IN OP CODE   */
                    /* ORDER.                                   */
                    /*                                          */
                    /*<><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                          */
          OP_DEGL(59) BIT(8) INITIAL(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 2, 2, 1, 2, 2
, 1, 0, 0, 0, 1, 2, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1),
                    /*                                          */
                    /*<><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                          */
                    /*              DEGREE OF RESULT            */
                    /* THE NUMBER OF RESULTS RESULTING FROM EACH*/
                    /* OP CODE.                                 */
                    /*                                          */
                    /*<><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                          */
          OP_DEGR(59) BIT(8) INITIAL(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0
, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0),
                    /*                                          */
                    /*<><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                          */
                    /*              TYPE POINTER                */
                    /* GIVES A DIFFERENCE BETWEEN CELLS THAT    */
                    /* ARE EQUAL TO DEGL + DEGR FOR OPERATOR.   */
                    /* INDEXED BY OP_CODE.  POINTS TO LOWER     */
                    /* OP_INFO TABLE TO FIND TYPES.             */
                    /*                                          */
                    /*<><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                          */
          OP_INDEX(60) BIT(8) INITIAL(0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3
, 5, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34, 37, 40, 43, 46, 49, 52, 55, 58, 61,
63, 65, 67, 70, 73, 75, 78, 80, 81, 82, 82, 82, 83, 85, 85, 86, 86, 88, 89, 90,
91, 92, 93, 93, 94, 95),
                    /*                                          */
                    /*<><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                          */
                    /*              OPERATOR INFORMATION        */
                    /* TYPE OF OPERANDS NECESSARY AND TYPE OF   */
                    /* RESULT NECESSARY FOR OPERATIONS.  EACH   */
                    /* LOCATION HOLDS A TYPE OR DEFAULT INDICATR*/
                    /* UP TO ENTRY 95.  ABOVE 95, THE TABLE     */
                    /* REVERTS TO SIMPLIFICATION DESCRIPTION.   */
                    /*                                          */
                    /*<><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                          */
          OP_INFO(170) BIT(16) INITIAL(0, 6, 5, 6, 5, 5, 6, 5, 5, 5, 6, 6, 6, 5, 5
, 5, 6, 6, 6, 5, 5, 5, 6, 6, 6, 5, 5, 5, 6, 6, 6, 5, 5, 5, 6, 5, 6, 5, 6, 6, 6,
6, 6, 1, 1, 7, 1, 1, 7, 1, 1, 7, 1, 1, 7, 1, 1, 7, 1, 1, 7, 5, 5, 6, 6, 7, 7, 7
, 7, 7, 7, 7, 7, 2, 1, 2, 1, 1, 2, 1, 1, 1, 4, 7, 4, 4, 5, 2, 2, 6, 5, 7, 1, 5,
1, 0, 3, 2, 2, 0, 3, 2, 2, 111, 3, 2, 3, 115, 3, 2, 3, 0, 5, 2, 2, 0, 5, 2, 2, 0
, 2, 2, 3, 0, 2, 2, 3, 0, 3, 2, 3, 0, 3, 2, 3, 151, 2, 3, 5, 155, 2, 3, 5, 159,
2, 3, 5, 163, 2, 3, 5, 0, 2, 5, 2, 0, 2, 5, 2, 0, 2, 5, 2, 167, 2, 5, 2, 0, 2, 5
2)
```

```
                /*                                          */
                /*<><><><><><><><><><><><><><><><><><><><><>*/
                /*                                          */
                /* POINTS TO SIMPLIFICATION DESCRIPTION IN   */
                /* OP_INFO BY OPERATOR.  OP_INFO ENTRIES ARE*/
                /* STRUCTURED AS:                            */
                /*         X = POINT TO NEXT RULE            */
                /*     X + 1 = OPERANDS (OP_DEGL)            */
                /*     X + 1 = RESULT (OP_DEGR)              */
                /* EACH LOCATION IS 8 BITS AS FOLLOWS:       */
                /*         BITS 7 - 1               0        */
                /*         ID# OR CONST#            1 = CONST*/
                /*                                  0 = ID   */
                /* ID    = MEANS WE USE VARIABLE VALUES      */
                /* CONST = SUBSCRIPT TO TABLE OP_STR TO GET */
                /*         IDENTITY CONSTANT IS CONST#.      */
                /* THE ONLY OPERATORS WHICH WILL RESULT IN   */
                /* A SIMPLIFICATION RULE ARE NUMBERED 16 TO */
                /* 27.  FOR EXAMPLE:                         */
                /*         1) ADDS                           */
                /*         2) SUBTRACTS                      */
                /*         3) EXPONENTIATION                 */
                /*         4) MULTIPLY                       */
                /*         5) DIVIDES                        */
                /*                                          */
                /*<><><><><><><><><><><><><><><><><><><><><>*/
                /*                                          */
         SIMP_INDEX(59)  BIT(8) INITIAL(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0,  95, 99, 119, 123, 103, 107, 127, 131, 135, 139, 143, 147, 0, 0, 0, 0, 0
, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0),
         (COLLEC,LOAD,STORE,STORET,DELETE,
         DUPLIC,CONV,COMM,EXCH,COND,UCOND,GOSUB,RETSUB)
         BIT(8),
                /*                                          */
                /*<><><><><><><><><><><><><><><><><><><><><>*/
                /*                                          */
                /*                 IDENTITY CONSTANTS        */
                /* THIS TABLE IS USED FOR IDENTITY CONSTANTS*/
                /* IN THE SIMPLIFICATION PROCEDURE.          */
                /* SUBSCRIPTS FOR THIS TABLE ARE FOUND IN    */
                /* TABLE OP_INFO.                            */
                /*                                          */
                /*<><><><><><><><><><><><><><><><><><><><><>*/
                /*                                          */
         OP_STR(2) CHARACTER INITIAL('', '0', '1'),
                /*<><><><><><><><><><><><><><><><><><><><><>*/
                /*                                          */
                /* THE FOLLOWING ARE DECLARATIONS FOR        */
                /*               EXEC STACK                  */
                /*               VALUE TABLE (EXPRESSIONS)   */
                /*               STATE STACK (CURRENT POOL)  */
                /*               ADDRESS TABLE               */
                /*               CONSTANT TABLE              */
                /*               INTERMEDIATE CODE BUFFER    */
                /*               BLOCKHEADER POINTERS        */
                /*               OPTIMIZATION TYPE TABLE     */
                /*                                          */
                /*<><><><><><><><><><><><><><><><><><><><><>*/
                /*                                          */
         HASHBASE          LITERALLY      '127',   /* CURRENT HASHBASE USED IN    */
                                                   /* VALUE TABLE STORAGE         */
         MAXVAL            LITERALLY      '1024',  /* LIMIT OF VALUE TABLE        */
         MAXEXEC           LITERALLY      '25',    /* LIMIT OF EXEC STACK         */
         FORMAL            LITERALLY      '1',     /* INDICATOR FORMAL IDENTIFIER*/
         COMPU             LITERALLY      '0',     /* INDICATOR FOR EXPRESSION    */
                                                   /* COMPUTATION                 */
         EXCON (MAXEXEC) BIT(1),                   /* CONSTANT INDICATOR          */
         EXTYPE(MAXEXEC) BIT(8),                   /* TYPE OF CONSTANT            */
         EXVAL#(MAXEXEC) BIT(16),                  /* EQVAL. CLASS NUMBER         */
         EXNAM#(MAXEXEC) BIT(16),                  /* NOT USED                    */
         EXADD(MAXEXEC)  BIT(16),                  /* POINT TO ADDRESS TABLE      */
         VTSPACE           LITERALLY      '1',     /* PRESET SPACE FOR VAR TABLE */
         VTAB(MAXVAL)      BIT(16),                /* EXPRESSION TABLE            */
         MAXSTATE          LITERALLY      '850',   /* LIMIT FOR THE STATE TABLE   */
         STATES(MAXSTATE) BIT(16),                 /* HOLDS POINTER TO CURRENT    */
                                                   /* STATE INFO FOR EACH BLOCK   */
         STLOC FIXED,                              /* STATE POINTER               */
         VTOPR                FIXED,               /* POINT TO TOP OF EXPRESSION */
                                                   /* TABLE                       */
         VTOP                 FIXED,               /* POINTER TO EXPRESSION STACK*/
         ETOP FIXED,                               /* POINTER TO EXEC STACK       */
         ETOPR FIXED,                              /* TOP OF EXEC STACK           */
         ADDLOCR FIXED,                            /* TOP OF ADDR TABLES ** USED */
                                                   /* TO SAVE STATE CONDITIONS    */
                                                   /* IN THE TOP OF ADDR TABLE    */
         EADR FIXED,                               /* BASE IN EXEC STACK OF FIRST*/
                                                   /* OPERAND                     */
```

```
      RESIND FIXED,                                    /* INDICATOR FOR WHERE RESULT */
                                                       /* VALUE NUMBER IS FOUND      */
      OPERTR FIXED,                                    /* OPERATOR                   */
      HASHADR FIXED,                                   /* COLLISION HASH ADDRESS     */
      MAXADD          LITERALLY      '850',            /* MAX LIMIT OF ADDRESS TABLE */
      ADDRESS(MAXADD) FIXED,                           /* INTERNAL ADDR NUMBER       */
      ADDTYPE(MAXADD) BIT(8),                          /* TYPE OF NUMBER             */
      ADDVAL(MAXADD)  BIT(16),                         /* EQVAL. CLASS NUMBER        */
      ADDCON(MAXADD)  BIT(16),                         /* POINT TO CONSTANT TAB      */
      ADDLOC          FIXED,                           /* POINT TO ADDRESS TABLE     */
      ADDNAM(MAXADD) BIT(16),                          /* IF ADDRESS ENTRY IS EQUAL  */
                                                       /* TO A COMPUTATION, THIS IS  */
                                                       /* THE POINTER TO VTAB        */
      MAXCON          LITERALLY      '50',             /* LIMIT FOR CONSTANTS        */
      CONSYM(MAXCON) CHARACTER,                        /*INTERNAL SYMBOL REPRESENTATI*/
      CONTYPE(MAXCON) BIT(8),                          /* TYPE OF CONSTANT..INT,ETC  */
      CONADD(MAXCON) BIT(16),                          /* CONSTANT ADDRESS           */
      CONVAL(MAXCON) BIT(16),                          /* EQVAL. CLASS NUMBER        */
      CONINT(MAXCON) FIXED,                            /* CONSTANT INTERNAL VALUE    */
      CONLOC FIXED,                                    /* POINTER TO CONSTANT TABS   */
      PASS#           FIXED,                           /* PASSES FOR CURRENT BLOCK   */
      #XITS           FIXED,                           /* EXITS CURRENT BLOCK        */
      VALUE_NUM_CNTR  FIXED,                           /* ACTUAL COUNTER USED TO     */
                                                       /* GENERATE VALUE NUMBERS     */
      CSF_FILE        LITERALLY      '1',              /* FILE # UPON WHICH THE      */
                                                       /* INTERMEDIATE LANGUAGE CAN  */
                                                       /* BE FOUND                   */
      DISKWORDS       LITERALLY      '900',            /* 3600 BYTES                 */
      MAXCODE         LITERALLY     '2700',            /* MAX CODE ALLOWED           */
      #PARMS          LITERALLY      '0',
      CODELOC FIXED,                                   /* POINTER TO CSFCODE         */
      CSFCODE(MAXCODE)FIXED,                           /* INPUT CODE/CONS AREA       */
      OPTIM_TYPE(MAXCODE) BIT(8),                      /* TABLE TO DECLARE TYPE OF   */
                                                       /* OPTIMIZATION FOUND:        */
      LASTCON FIXED,                                   /* POINT TO THE LAST CONSTANT */
      MAXCONT         LITERALLY      '50',             /* LIMIT FOR CONTROL TABLE    */
      CONTROL(MAXCONT) FIXED,                          /* BLK# - BLK FWA ** THIS IS  */
                                                       /* THE LIST FOR THE TABLE     */
                                                       /* PROCESSED BLOCKS.          */
      SAV_TOG(MAXCONT) BIT(8),                         /* TOGGLES FOR THIS BLOCK     */
      SELECT_FLAG FIXED,                               /* SET TO INDICATE TYPE OF    */
                                                       /* SELECTION METHOD DESIRED   */
                                                       /* FROM CONTROL LIST:         */
      MAXBLK          LITERALLY      '90',             /* MAXIMUM # OF BLOCKS IN A   */
                                                       /* PROGRAM                    */
      BLK# FIXED,                                      /* HOLDS CURRENT BLK#         */
      BLKCNT FIXED,                                    /* # OF BLOCKS                */
      BLKLOC FIXED,                                    /* POINTS TO LWA + 1 OF THE   */
                                                       /* BLOCK HEADER AREA          */
      BLKPNT(MAXBLK) FIXED,                            /* HOLDS FWA IN CSFCODE FOR   */
                                                       /* EACH BLOCK HEADER          */
      BLKSIZE         LITERALLY      '4',              /* PRESET MINIMUM SIZE OF     */
                                                       /* BLOCK HEADER               */
      M_CONTROL FIXED,                                 /* POINT TO CSF FILE FOR FLOW */
      CTOP FIXED,                                      /* CONTROL STACK POINTER      */
      CURBLOCK FIXED,                                  /* HOLDS ADDRESS OF CURRENT   */
                                                       /* BLOCK HEADER               */
      MAX_OPTIMIZATIONS LITERALLY    '100',            /* LIMIT ON THE NUMBER OF     */
                                                       /* OPTIMIZATIONS THAT CAN BE  */
                                                       /* FOUND.                     */
      OPTIM_ADR_LIST(MAX_OPTIMIZATIONS) FIXED,/* HOLDS THE OPTIMIZATION EXP */
                                                       /* BLOCK ADDRESS IN THE TOP   */
                                                       /* OF VTAB FOR EVERY POSSIBLE */
                                                       /* OPTIMIZATION.              */
      OPTOP           FIXED,                           /* POINTER TO OPTIM_ADR_LIST  */
              /*                                       */
              /*<><><><><><><><><><><><><><><><><><><>*/
              /*                                       */
              /*          MISC VARIABLES               */
              /*                                       */
              /*<><><><><><><><><><><><><><><><><><><>*/
              /*                                       */
      FIRSTCALL BIT(1),                                /* USED IN PERMUTE TO INDICATE*/
                                                       /* THE FIRST PASS FOR A STRING*/
      BOPA FIXED,                                      /* FIRST OPERAND POINTER      */
      BOPB FIXED,                                      /* SECOND OPERAND POINTER     */
      TADDLOC FIXED,                                   /* CURRENT BOTTOM OF ADDRESS  */
                                                       /* TABLE FOR BLOCK BEING      */
                                                       /* PROCESSED                  */
      NUM_SELECT_METHODS    LITERALLY '3',             /* MAX NUMBER FOR BLOCK SELECT*/
                                                       /* CODE                       */
      STRTIME FIXED,                                   /* HOLDS THE TIME FOR START OF*/
                                                       /* OPTIM                      */
      PAUTIME FIXED,                                   /* HOLDS COMPLETION TIME FOR  */
                                                       /* OPTIM                      */
      TOTIME FIXED,                                    /* HOLDS TOTAL TIME FOR OPTIM */
                                                       /* IN CENTISECONDS            */
      SECONDS FIXED,                                   /* HOLDS SECONDS FOR OPTIM    */
      PRTSEC FIXED,                                    /* HOLDS PARTIAL SECONDS FOR  */
                                                       /* OPTIM                      */
```

```
TRUE BIT(1) INITIAL("1"),
FALSE BIT(1) INITIAL("0"),
RIGHT BIT(1) INITIAL("1"),
LEFT BIT(1) INITIAL("0"),
PAGEJT CHARACTER INITIAL('1'),             /* USED TO PAGE EJECT          */
SINGLE CHARACTER INITIAL (' '),            /* USED TO SINGLE SPACE        */
SINGLE_SPACE LITERALLY 'OUTPUT(1) = SINGLE',
PAGE_EJECT LITERALLY 'OUTPUT(1) = PAGEJT',
      /*<><><><><><><><><><><><><><><><><><><><><><><>*/
      /*                                              */
      /* DECLARATIONS FOR THE CSF TRACE ROUTINES      */
      /*                                              */
      /*<><><><><><><><><><><><><><><><><><><><><><><>*/
#TOGS            LITERALLY      '5',             /* # OF PRINT TOGGLES     */
TOGGLES (#TOGS)  BIT(1),
TRACETOG         LITERALLY      'TOGGLES(0)',    /*SET IF TRACE REQUESTED*/
TABLETOG         LITERALLY      'TOGGLES(1)',    /* SET IF TABLES REQUEST*/
PROPTOG          LITERALLY      'TOGGLES(2)',    /* SET IF PROPAGATION   */
BRANCHTOG        LITERALLY      'TOGGLES(3)',    /* SET IF MULTIPLE EXIT */
PUNCHTOG         LITERALLY      'TOGGLES(4)',    /* SELECT PUNCH CARDS   */
COL# FIXED,
T_WIDTH          LITERALLY      '10',      /* SIZE OF HDR TITLE           */
N_COLS           LITERALLY      '14',      /* NUMBER OF COLUMNS           */
C_WIDTH          LITERALLY      '5',       /* SIZE OF COLUMNS             */
BLANKS      CHARACTER ,                    /* USED TO FORCE SPACES        */
TBUFF       CHARACTER ,                    /* BUFFER FOR OUTPUT           */
HEADING     CHARACTER ,                    /* BUFFER FOR HEADERS          */
#EXF             LITERALLY      '3',       /* # OF FIELDS FOR EXEC HEAD   */
#CNF             LITERALLY      '3',       /* # OF FIELDS FOR CON HEAD    */
#ADF             LITERALLY      '3',       /* # OF FIELDS FOR ADR HEAD    */
#CTF             LITERALLY      '1',       /* # FIELDS FOR CONTROL HEADER*/
                                           /*                             */
                                           /* BLOCK SELECTION METHOD      */
                                           /* MESSAGES.                   */
                                           /*      0 = LIFO               */
                                           /*      1 = STEEPEST DESCENT   */
                                           /*      2 = FIFO               */
                                           /*      3 = THE BLOCK WITH THE */
                                           /*          MAXIMUM CURRENT    */
                                           /*          POOL IS SELECTED   */
                                           /*                             */
SELECT_METHOD (4) CHARACTER INITIAL ('LAST-IN-FIRST-OUT ',
    'STEEPEST DESCENT (MINIMUM CURRENT POOL) ',
    'FIRST-IN-FIRST-OUT ',
    'MAXIMUM CURRENT POOL '),
                                           /*                             */
                                           /* TABLE FOR OPTIMIZATION TYPE*/
                                           /* MESSAGES.  CURRENT TYPES    */
                                           /* ARE:                        */
                                           /*      0 = NONE               */
                                           /*      1 = SIMPLIFICATION RULE */
                                           /*          APPLIES            */
                                           /*      2 = CONSTANT PROPAGATION*/
                                           /*      3 = COMMON SUBEXPRESSION*/
                                           /*          ELIMINATION        */
                                           /*      4 = COMMON SUBEXPRESSION*/
                                           /*          THAT LINKS TO A    */
                                           /*          CONSTANT PROPAGATION*/
                                           /*                             */
TYPOPTIM (5) CHARACTER INITIAL(' ', 'SIMPLIFICATION',
    'CONSTANT PROPAGATION', 'COMM SUBEXP ELIM',
    'COMM SUBEXP ELIM & CONS PROP'),
                                           /*                             */
                                           /* EXEC STACK PRINT HEADINGS   */
                                           /*                             */
EXHEAD (#EXF) CHARACTER INITIAL('EXCON','EXTYPE','EXADD',
    'EXVAL#'),
                                           /*                             */
                                           /* ADDR TAB PRINT HEADINGS     */
                                           /*                             */
ADHEAD(#ADF) CHARACTER INITIAL('ADDRESS','ADDTYPE','ADDVAL',
    'ADDCON'),
                                           /*                             */
                                           /* CONSTANT TABLE PRINT HEAD   */
                                           /*                             */
CNHEAD (#CNF) CHARACTER INITIAL('CONSYM','CONTYPE','CONVAL',
    'CONINT'),
                                           /*                             */
                                           /* CONTROL STACK PRINT HEADING*/
                                           /*                             */
CTHEAD (#CTF) CHARACTER INITIAL('BLOCK#','ENTRY'),
                                           /* TOGGLES TO INDICATE THAT    */
                                           /* VARIOUS TABLE DUMPS ARE     */
                                           /* REQUESTED WHEN W_ALL IS     */
                                           /* CALLED.                     */
ALTEX            BIT(1),                   /* EXEC STACK DUMP             */
ALTAD            BIT(1),                   /* ADDRESS TABLE DUMP          */
ALTCN            BIT(1),                   /* CONSTANT TABLE DUMP         */
ALTST            BIT(1),                   /* STATE TABLE DUMP            */
ALTCT            BIT(1),                   /* CONTROL STACK DUMP          */
ALTVL            BIT(1);                   /* RAW VALUE TABLE DUMP        */
```

```
                    /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                              */
                    /*                LINE_FEED                     */
                    /* ROUTINE SIMPLY RESULTS IN A LINE FEED        */
                    /* IN THE OUTPUT, IE A BLANK LINE.              */
                    /*                                              */
                    /*<><><><><><><><><><><><><><><><><><><><><><><>*/
LINE_FEED: PROCEDURE (LF#);
     DECLARE (LF#, COUNT) FIXED;
     COUNT = 0;
     DO WHILE COUNT < LF#;
          SINGLE_SPACE;
          IF PUNCHTOG THEN
          OUTPUT(2) = ' ';
          COUNT = COUNT + 1;
     END;
END LINE_FEED;
                    /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                              */
                    /*                      ERROR                   */
                    /* OUTPUTS ERROR MESSAGE OF M AND ERROR         */
                    /* NUMBER OF I                                  */
                    /* ERRORS ARE:                                  */
                    /*         NUMBER          CAUSE                */
                    /*            1       VARIOUS STACK OVERFLOWS   */
                    /*            2       ERROR IN ADDRESSING OF    */
                    /*                    OF VARIOUS TABLES.        */
                    /*            3       MATHEMATICAL OPERATION    */
                    /*                    REQUESTED IS INCORRECT    */
                    /*            4       INPUT CODE FILE EXCEEDS   */
                    /*                    MAX PROGRAM SIZE          */
                    /*            5       BLOCK SELECTION CODE      */
                    /*                    REQUESTED IS TOO LARGE    */
                    /*            6       TRIED TO DELETE A         */
                    /*                    NON-EXISTENT VALUE #      */
                    /*                    FROM THE CLASS OCCURANCE*/
                    /*                    LIST IN PERFORM_MEET_OP.*/
                    /*            7       THE OPTIMIZATION ADDRESS*/
                    /*                    TABLE HAS OVERFLOWED.     */
                    /*                    PRESENTLY THE PGM CAN     */
                    /*                    HANDLE UP TO 100          */
                    /*            -       OPTIMIZATIONS.            */
                    /*           10       COMPILATION ERROR         */
                    /* ROUTINES CALLED:                             */
                    /*              LINE_FEED                       */
                    /*                                              */
                    /*<><><><><><><><><><><><><><><><><><><><><><><>*/
ERROR: PROCEDURE (M,I);
     DECLARE M CHARACTER, I FIXED;
     OUTPUT = ' *** '||M||' ***   ERROR TYPE :=  ' || I;
     IF PUNCHTOG THEN
     OUTPUT(2) = ' *** '||M||' ***   ERROR TYPE :=  ' || I;
     CALL LINE_FEED(2);
END ERROR;
                    /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                              */
                    /*                   PAD                        */
                    /* THIS ROUTINE PADS A STRING WITH SPACES       */
                    /* TO A SPECIFIED LENGTH (SPACES ADDED TO       */
                    /* RIGHT OR LEFT AS DETERMINED BY DIR IND.      */
                    /*           0 = LEFT                           */
                    /*           1 = RIGHT                          */
                    /*                                              */
                    /*<><><><><><><><><><><><><><><><><><><><><><><>*/
PAD: PROCEDURE(STRING, SIZE, DIR) CHARACTER;
     DECLARE(T, STRING) CHARACTER, SIZE FIXED, DIR BIT(1);
     IF DIR THEN
     DO WHILE LENGTH(STRING) < SIZE;
          STRING = STRING || ' ';
     END;
     ELSE
     DO WHILE LENGTH(STRING) < SIZE;
          T = ' ';
          T = T || STRING;
          STRING = T;
     END;
     RETURN SUBSTR(STRING, 0, SIZE);
END PAD;
                    /*<><><><><><><><><><><><><><><><><><><><><><><>*/
```

```
                    /*                                          */
                    /*                      HEX                 */
                    /* CONVERTS I TO HEXIDECIMAL FORMAT AND     */
                    /* RETURNS IT AS A STRING.                  */
                    /*                                          */
                    /*<><><><><><><><><><><><><><><><><><><><>*/
HEX: PROCEDURE (INVAL) CHARACTER;
     DECLARE (INVAL, NUMDIG, DIG) FIXED, T CHARACTER;
     T = '';
     DO NUMDIG = 0 TO 7;
          DIG = SHR(INVAL,SHL(NUMDIG, 2)) & "F";
          T = SUBSTR('0123456789A3CDEF', DIG, 1) || T;
     END;
     RETURN T;
END HEX;
                    /*<><><><><><><><><><><><><><><><><><><><>*/
                    /*                                          */
                    /*             W_CLEAR                      */
                    /*  OUTPUTS TBUFF AND THEN SETS IT TO NULL  */
                    /*  AND COLUMN COUNTER TO ZERO.  TBUFF WILL */
                    /*  NOT BE OUTPUT IF IT IS NULL.            */
                    /*                                          */
                    /*<><><><><><><><><><><><><><><><><><><><>*/
W_CLEAR: PROCEDURE;
     IF LENGTH(TBUFF) > 0 THEN
     DO;
          OUTPUT = TBUFF;
          IF PUNCHTOG THEN
          OUTPUT(2) = TBUFF;
     END;
     TBUFF = '';
     COL#=0;
END W_CLEAR;
                    /*<><><><><><><><><><><><><><><><><><><><>*/
                    /*                                          */
                    /*             W_TITLE                      */
                    /*  OUTPUTS STRING T AS A TITLE LINE.       */
                    /*  THE TITLE WIDTH IS SET BY T-WIDTH WHICH */
                    /*  IN TURN MARKS THE COLUMN COUNTER SO THE */
                    /*  REMAINDER OF THE TITLE LINE CAN BE      */
                    /*  BUILT.                                  */
                    /*  ROUTINES CALLED:                        */
                    /*             PAD                          */
                    /*                                          */
                    /*<><><><><><><><><><><><><><><><><><><><>*/
W_TITLE: PROCEDURE (T);
     DECLARE T CHARACTER, I FIXED;
     IF LENGTH(T) > T_WIDTH THEN
     T = SUBSTR(T, 0, T_WIDTH);
     ELSE
     T = PAD(T,T_WIDTH,RIGHT);
     COL#=T_WIDTH;
     DO I = 1 TO N_COLS;
          COL# = COL# + C_WIDTH;
          T = T ||' <' ||I|| '>';
          T = PAD(T,COL#,RIGHT);
     END;
     OUTPUT = T;
     IF PUNCHTOG THEN
     OUTPUT(2) = T;
     COL# = 0;
END W_TITLE;
                    /*<><><><><><><><><><><><><><><><><><><><>*/
                    /*                                          */
                    /*          READ_SELECT_METHOD              */
                    /* THIS ROUTINE INPUTS THE BLOCK SELECT     */
                    /* METHOD CODE:                             */
                    /*     0 = LIFO                             */
                    /*     1 = STEEPEST DESCENT                 */
                    /*     2 = FIFO                             */
                    /*     3 = FIFO AND THEN STEEPEST DESCENT   */
                    /*     4 = LIFO AND THEN STEEPEST DESCENT   */
                    /*     5 = BLOCK WITH MAXIMUM CURRENT POOL  */
                    /*         SELECTED                         */
                    /*                                          */
                    /*<><><><><><><><><><><><><><><><><><><><>*/
READ_SELECT_METHOD: PROCEDURE FIXED;
     DECLARE TEMA FIXED;
     TBUFF = INPUT;
     TEMA = BYTE(TBUFF, 0);
     TEMA = TEMA - "F0";
     TBUFF = '';
     RETURN TEMA;
END READ_SELECT_METHOD;
```

```
                    /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                               */
                    /*                   W_ITEM                       */
                    /*  PUTS STRING T BEGINNING AT COLUMN N           */
                    /*  INTO TBUFF.  IF TBUFF IS FILLED AS A          */
                    /*  RESULT OF THE OPERATION TBUFF IS OUTPUT.*/
                    /*  ROUTINES CALLED:                              */
                    /*               PAD                              */
                    /*                                               */
                    /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
W_ITEM: PROCEDURE (N,T);
     DECLARE (I,N)FIXED, T CHARACTER;
     IF COL#=0 THEN
     DO;
          IF N=1 THEN
          DO;
               TBUFF = ' ' || HEADING;
               IF LENGTH(TBUFF) > T_WIDTH THEN
               TBUFF = SUBSTR(TBUFF, 0, T_WIDTH);
          END; ELSE
          DO;
               I = N / N_COLS;
               I = (I + 1) * N_COLS;
               TBUFF='     '||I||'+';
          END;
          TBUFF = PAD(TBUFF,T_WIDTH,RIGHT);
          COL# = T_WIDTH;
     END;
     IF LENGTH(T) >= C_WIDTH THEN
     T = SUBSTR(T, 0, C_WIDTH - 1);
     COL# = COL# + C_WIDTH;
     TBUFF = TBUFF || '|' || T;
     TBUFF = PAD(TBUFF,COL#,RIGHT);
     I=N MOD N_COLS;
     IF I=0 THEN
     DO;
          OUTPUT = TBUFF;
          IF PUNCHTOG THEN
          OUTPUT(2) = TBUFF;
          TBUFF = ' ';
          COL# = 0;
     END;
END W_ITEM;
                    /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                               */
                    /*                  W_NUM                         */
                    /* PUTS NUMBER AT ADDRESS IN COLUMN OF TBUFF*/
                    /* ROUTINES CALLED:                               */
                    /*               W_ITEM                           */
                    /*                                               */
                    /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
W_NUM: PROCEDURE (COL, ADR);
     DECLARE (COL, ADR) FIXED, T CHARACTER;
     T = ADR;
     CALL W_ITEM(COL, T);
END W_NUM;
                    /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                               */
                    /*                   W_TYPE                       */
                    /*  PUTS OPERATION CODE SPECIFIED BY I AND        */
                    /*  J IN COLUMN N OF TBUFF.  I AND J ARE          */
                    /*  POINTERS TO THE OPRANGE TABLE WHICH           */
                    /*  SIMPLY IS FWA/LWA FOR ADDRESSING THE          */
                    /*  OPCODES TABLE.                                */
                    /*  ROUTINES CALLED:                              */
                    /*               W_ITEM                           */
                    /*                                               */
                    /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
W_TYPE: PROCEDURE (N,I);
     DECLARE (I,J,N) FIXED;
     J = OPRANGE(I);
     I = OPRANGE(I + 1) - J;
     CALL W_ITEM(N,SUBSTR(OPCODES,J,I));
END W_TYPE;
                    /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                               */
                    /*                      W_CON                     */
                    /*  PUTS Y OR N DEPENDING ON C (TRUE/FALSE) */
                    /*  IN COLUMN N OF TBUFF.  THE Y OR N             */
                    /*  SIMPLY THE CONSTANT INDICATOR.                */
                    /*  ROUTINES CALLED:                              */
                    /*               W_ITEM                           */
                    /*                                               */
                    /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
W_CON: PROCEDURE (N,C);
     DECLARE (N,C) FIXED;
     IF C THEN
     CALL W_ITEM(N, 'Y');
     ELSE CALL W_ITEM(N, 'N');
END W_CON;
```

```
                    /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                                */
                    /*              W_EXEC                            */
                    /* PRINT THE EXEC STACK TABLES                    */
                    /* ROUTINES CALLED:                               */
                    /*              W_TITLE                           */
                    /*              LINE_FEED                         */
                    /*              W_CON                             */
                    /*              W_TYPE                            */
                    /*              W_NUM                             */
                    /*              W_CLEAR                           */
                    /*                                                */
                    /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
W_EXEC: PROCEDURE;
    DECLARE (I,J) FIXED;
    ALTEX=0;
    CALL W_TITLE(' EXEC STACK');
    IF ETOP < 1 THEN
    DO;
        OUTPUT = ' (EMPTY)';
        IF PUNCHTOG THEN
        OUTPUT(2) = ' (EMPTY)';
        CALL LINE_FEED(2);
        RETURN;
    END;
    DO I = 0 TO #EXF;
        HEADING=EXHEAD(I);
        DO J = 1 TO ETOP;
            DO CASE I;
                CALL W_CON(J,EXCON(J));
                CALL W_TYPE(J,EXTYPE(J));
                CALL W_NUM(J,EXADD(J));
                CALL W_NUM(J,EXVAL#(J));
            END;
        END;
        CALL W_CLEAR;
    END;
    CALL LINE_FEED(2);
END W_EXEC;
                    /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                                */
                    /*              W_ADD                             */
                    /* PRINTS THE ADDRESS TABLES.                     */
                    /* ROUTINES CALLED:                               */
                    /*              W_TITLE                           */
                    /*              LINE_FEED                         */
                    /*              W_NUM                             */
                    /*              W_TYPE                            */
                    /*              W_CLEAR                           */
                    /*                                                */
                    /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
W_ADD: PROCEDURE;
    DECLARE (I, J) FIXED;
    ALTAD=0;
    CALL W_TITLE(' ADDR TABLE');
    IF ADDLOC < 1 THEN
    DO;
        OUTPUT = ' (EMPTY)';
        IF PUNCHTOG THEN
        OUTPUT(2) = ' (EMPTY)';
        CALL LINE_FEED(2);
        RETURN;
    END;
    DO I = 0 TO #ADF;
        HEADING=ADHEAD(I);
        DO J = 1 TO ADDLOC;
            DO CASE I;
                CALL W_NUM(J,ADDRESS(J));
                CALL W_TYPE(J,ADDTYPE(J));
                CALL W_NUM (J,ADDVAL(J));
                CALL W_NUM (J,ADDCON(J));
            END;
        END;
        CALL W_CLEAR;
    END;
    CALL LINE_FEED(2);
END W_ADD;
```

```
                /*<><><><><><><><><><><><><><><><><><><><><><>*/
                /*                                           */
                /*            W_CONS                         */
                /* OUTPUTS THE CONSTANT TABLES.              */
                /* ROUTINES CALLED:                          */
                /*            W_TITLE                        */
                /*            LINE_FEED                      */
                /*            W_ITEM                         */
                /*            W_TYPE                         */
                /*            W_NUM                          */
                /*            W_CLEAR                        */
                /*                                           */
                /*<><><><><><><><><><><><><><><><><><><><><><>*/
W_CONS: PROCEDURE;
    DECLARE (I,J) FIXED;
    ALTCN=0;
    CALL W_TITLE(' CONSTANTS');
    IF CONLOC < 1 THEN
    DO;
        OUTPUT = ' (EMPTY)';
        IF PUNCHTOG THEN
        OUTPUT(2) = ' (EMPTY)';
        CALL LINE_FEED(2);
        RETURN;
    END;
    DO I = 0 TO #CNF;
        HEADING=CNHEAD(I);
        DO J = 1 TO CONLOC;
            DO CASE I;
                CALL W_ITEM(J,CONSYM(J));
                CALL W_TYPE(J,CONTYPE(J));
                CALL W_NUM (J,CONVAL(J));
                CALL W_NUM (J,CONINT(J));
            END;
        END;
        CALL W_CLEAR;
    END;
    CALL LINE_FEED(2);
END W_CONS;
                /*<><><><><><><><><><><><><><><><><><><><><><>*/
                /*                                           */
                /*            W_CONT                         */
                /* OUTPUTS THE CONTROL STACK.                */
                /* ROUTINES CALLED:                          */
                /*            W_TITLE                        */
                /*            LINE_FEED                      */
                /*            W_ITEM                         */
                /*            W_NUM                          */
                /*            W_CLEAR                        */
                /*                                           */
                /*<><><><><><><><><><><><><><><><><><><><><><>*/
W_CONT: PROCEDURE;
    DECLARE (I,J) FIXED;
    ALTCT=0;
    CALL W_TITLE(' CONT STK');
    IF CTOP < 1 THEN
    DO;
        IF PUNCHTOG THEN
        OUTPUT(2) = ' (EMPTY)';
        OUTPUT = ' (EMPTY)';
        CALL LINE_FEED(2);
        RETURN;
    END;
    DO I = 0 TO #CTF;
        HEADING=CTHEAD(I);
        DO J = 1 TO CTOP;
            DO CASE I;
                IF SHR(CONTROL(J),31) THEN
                CALL W_ITEM(J,'*'); ELSE
                CALL W_NUM(J,SHR(CONTROL(J),16));
                CALL W_NUM(J,CONTROL(J)&"FFFF");
            END;
        END;
        CALL W_CLEAR;
    END;
    CALL LINE_FEED(2);
END W_CONT;
```

```
                    /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                                */
                    /*                   W_VAL                        */
                    /* OUTPUTS THE VALUE STACK IN THE FORMAT          */
                    /*          C(X Y Z | W)                          */
                    /*          X = OPERATOR                          */
                    /*          Y = CLASS # OPERAND A                 */
                    /*          Z = CLASS # OPERAND B                 */
                    /*          W = CLASS # FOR COMPUTATION           */
                    /* ROUTINES CALLED:                               */
                    /*                   W_CLEAR                      */
                    /*                   LINE_FEED                    */
                    /*                                                */
                    /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
W_VAL: PROCEDURE;
    DECLARE (I,J,N,K,L,M) FIXED , T CHARACTER;
    ALTVL=0;
    TBUFF = ' ';
    N = T_WIDTH + N_COLS * C_WIDTH;
    OUTPUT = ' VALUE STACK (TOP AT '||VTOP||')';
    IF PUNCHTOG THEN
    OUTPUT(2) =' VALUE STACK (TOP AT '||VTOP||')';
    I = VTOP;
    J = VTAB(VTOP);
    DO WHILE J > 0;
        K = VTAB(J + 1);
        IF SHR(K, 15) THEN T = 'F';
        ELSE T = 'C';
        M = K & "FF";
        L = OPRANGE(M);
        K = OPRANGE(M + 1) - L;
        T=T||'('||SUBSTR(OPCODES,L,K);
        K = OP_DEGL(M);
        L = J + 3;
        DO WHILE K > 0;
            K = K - 1;
            T = T || ' ' || VTAB(L);
            L = L + 1;
        END;
        T=T||' |'||VTAB(L)||') ';
        I = J;
        J = VTAB(J);
        IF LENGTH(TBUFF) + LENGTH(T) > N THEN
        DO;
            OUTPUT = TBUFF;
            IF PUNCHTOG THEN
            OUTPUT(2) = TBUFF;
            TBUFF = T;
        END;
        ELSE TBUFF = TBUFF || T;
    END;
    CALL W_CLEAR;
    CALL LINE_FEED(2);
END W_VAL;
                    /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                                */
                    /*                   W_STATE                      */
                    /* OUTPUTS THE STATE CORRESPONDING TO             */
                    /* BLOCK AT ADDRESS CB IN CSFCODE.                */
                    /* ROUTINES CALLED:                               */
                    /*                   LINE_FEED                    */
                    /*                   W_CLEAR                      */
                    /*                                                */
                    /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
W_STATE: PROCEDURE (CB, WHERE);
    DECLARE (I, K, J, M, CB, X, TEMP, NUMOP) FIXED,
            (T, WHERE) CHARACTER;
    CALL LINE_FEED(1);
    ALTST = 0;
    IF CB = 0 THEN RETURN;
    M=T_WIDTH+N_COLS*C_WIDTH;
    I = CSFCODE(CB);
    K = SHR(I,16);
    I = I & "FFFF";
    TBUFF = ' ';
    OUTPUT = ' ' || WHERE || ' BLOCK# ' || K || ' IS: ';
    IF PUNCHTOG THEN
    OUTPUT(2) = ' '||WHERE ||' BLOCK# '|| K || ' IS: ';
    CB=STATES(I+1);
                                                        /*                           */
                                                        /* THERE ARE TWO EMPTY SPACES*/
                                                        /* AT THE BEGINNING OF THE   */
                                                        /* STATE VECTOR.             */
                                                        /*                           */
    I = I + 2;
    J = STATES(I);
                                                        /*                           */
                                                        /* IS AN OPTIMIZED STATE     */
                                                        /* PRESENT                   */
                                                        /*                           */
```

```
      IF J = 0 THEN
                                                      /*                              */
                                                      /* NO...INDICATE EMPTY          */
                                                      /*                              */
      DO;
         OUTPUT = ' (NULL)';
         IF PUNCHTOG THEN
         OUTPUT(2) = ' (NULL)';
      END;
      ELSE DO;
                                                      /*                              */
                                                      /* YES...PROCESS EACH           */
                                                      /* EQUIVALENCE CLASS            */
                                                      /*                              */
         DO WHILE J > 0;
            J = J - 1;
            I = I + 1;
            K = STATES(I) & "7FFF";
            X = ADDRESS(K);
            IF SHR(X, 15) THEN                        /* EXEC STACK ELEMENT           */
                                                      /*      E ( X, Y, Z )           */
                                                      /*    X = RELATIVE LOCATION     */
                                                      /*        FROM THE TOP OF       */
                                                      /*        EXEC STACK            */
                                                      /*    Y = IF RESULT IS          */
                                                      /*        PROPAGATED CONSTANT*/
                                                      /*        VALUE IS CLASS #      */
                                                      /*    Z = CLASS NUMBER OF THE*/
                                                      /*        COMPUTATION RESULT */
            DO;
               T = ' E(';
               X = X & "7FFF";
            END; ELSE
            IF X = 0 THEN                             /* VALUE TABLE ENTRY            */
                                                      /* INVOLVING AT LEAST ONE       */
                                                      /* COMPUTATION.                 */
            DO;                                       /*      C( X,Y,Z )              */
               T = ' C(';
                                                      /*      OPERANDS                */
                                                      /*    Y = VALUE# OF THE         */
                                                      /*        PROPAGATED CONSTANT*/
                                                      /*        IF ONE EXISTS         */
                                                      /*    Z = CLASS NUMBER OF       */
                                                      /*        COMPUTATION RESULT */
               TEMP = ADDNAM(K);
               NUMOP = ADDTYPE(K) & "FF";
               NUMOP = OP_DEGL(NUMOP) - 1;
               DO WHILE NUMOP > 0;
                  NUMOP = NUMOP - 1;
                  X = VTAB(TEMP);
                  T = T || X || ':';
                  TEMP = TEMP - 1;
               END;
               X = VTAB(TEMP);
            END; ELSE
                                                      /*                              */
                                                      /* ADDRESS TABLE ENTRY          */
                                                      /*      A ( X, Y, Z )           */
                                                      /*    X = SIMPLE ID NUMBER      */
                                                      /*    Y = SAME AS ABOVE         */
                                                      /*    Z = SAME AS ABOVE         */
                                                      /*                              */
            T = ' A(';
            IF X = 0 THEN
            T = T || '*,';
            ELSE
            T = T || X || ',';
            X = ADDCON(K) & "7FFF";
            IF X = 0 THEN
            T = T || '*,';
            ELSE
            DO;
               X = CONVAL(X);
               T = T || X || ',';
            END;
            X = ADDVAL(K);
            IF X = 0 THEN
            T = T || '*) ';
            ELSE T = T || X || ') ';
            IF LENGTH(TBUFF) + LENGTH(T) > M THEN
            DO;
               OUTPUT = TBUFF;
               IF PUNCHTOG THEN
               OUTPUT(2) = TBUFF;
               TBUFF = ' '|| T;
            END;
            ELSE TBUFF = TBUFF || T;
         END;
      END;
      CALL W_CLEAR;
      CALL LINE_FEED(1);
   END W_STATE;
```

```
              /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
              /*                                                */
              /*               PRINT_OUTPUT_POOL                */
              /* PRINTS THE OUTPUT POOL FROM THE CURRENT        */
              /* DATA ON THE ADDRESS STACK IN THE SAME          */
              /* MANNER AS W_STATE PRINTS THE CURRENT           */
              /* POOL.                                          */
              /* ROUTINES CALLED:                               */
              /*               ERROR                            */
              /*               W_CLEAR                          */
              /*               LINE_FEED                        */
              /*                                                */
              /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
PRINT_OUTPUT_POOL: PROCEDURE;
    DECLARE (CURADR, IDENTOP, ID, DATA, NUMOP, TEMP, MAXWIDTH, I,
            TEMPA) FIXED,
            MAXIDENTS LITERALLY '50',
            IDENT(MAXIDENTS) FIXED,
            T CHARACTER,
           (PRESENT, EXPRES, FOUND) BIT(1);
    MAXWIDTH = T_WIDTH + N_COLS * C_WIDTH;
    CURADR = ADDLOC;
    TBUFF = ' ';
    OUTPUT = '  OUTPUT POOL FROM BLOCK# '||BLK#|| ' IS: ';
    IF PUNCHTOG THEN
    OUTPUT(2) = '  OUTPUT POOL FROM BLOCK# '||BLK#|| ' IS: ';
    IF CURADR < 1 THEN
    DO;
        OUTPUT = ' (NULL)';
        IF PUNCHTOG THEN
        OUTPUT(2) = ' (NULL)';
        RETURN;
    END;
    IDENTOP = 1;
    EXPRES = FALSE;
    DO WHILE CURADR > 0;
        FOUND = FALSE;
        DATA, ID = ADDRESS(CURADR);
        IF SHR(ID, 15) THEN                          /* EXEC STACK ENTRY             */
        DO;
            FOUND = TRUE;
            T = ' E(';
            DATA = DATA & "7FFF";
        END;
        IF ID = 0 | EXPRES THEN
        DO;
            FOUND = TRUE;
            IF EXPRES THEN
            TEMP = VTAB(TEMPA) + 2;
                                                     /*                              */
                                                     /* EXPRESSION ENTRY             */
                                                     /*                              */
            ELSE
            TEMP = ADDNAM(CURADR) + 2;
            TEMPA = TEMP - 1;
            IF VTAB(TEMPA) ¬= 0 THEN
            EXPRES = TRUE;
            ELSE
            EXPRES = FALSE;
            T = ' C(';
            NUMOP = ADDTYPE(CURADR) & "FF";
            NUMOP = OP_DEGL(NUMOP) - 1;
            DO WHILE NUMOP > 0;
                NUMOP = NUMOP - 1;
                DATA = VTAB(TEMP);
                T = T || DATA || ':';
                TEMP = TEMP + 1;
            END;
            DATA = VTAB(TEMP);
        END; ELSE                                    /* ADDRESS STACK ENTRY          */
        DO;
            PRESENT = FALSE;
            I = IDENTOP - 1;
            DO WHILE (I > 0) & (¬ PRESENT);
                IF IDENT(I) = DATA THEN
                PRESENT = TRUE;
                I = I - 1;
            END;
            IF ¬PRESENT THEN
            DO;
                IDENT(IDENTOP) = DATA;
                IDENTOP = IDENTOP + 1;
                IF IDENTOP > MAXIDENTS THEN
                DO;
                    CALL ERROR('IDENTIFIER TABLE OVERFLOW', 1);
                    IDENTOP = IDENTOP - 1;
                END;
                FOUND = TRUE;
                T = ' A(';
            END;
```

```
        END;
        IF DATA = 0 THEN
        T = T || '*,';
        ELSE
        T = T || DATA || ',';
                                                                /*                          */
                                                                /* GET THE PROPAGATED CONSTAN*/
                                                                /* VALUE NUMBER             */
                                                                /*                          */
        DATA = ADDCON(CURADR) & "7FFF";
        IF DATA = 0 THEN
        T = T || '*,';
        ELSE
        DO;
            DATA = CONVAL(DATA);
            T = T || DATA || ',';
        END;
                                                                /*                          */
                                                                /* GET THE CURRENT VALUE #  */
                                                                /*                          */
        DATA = ADDVAL(CURADR);
        IF DATA = 0 THEN
        T = T || '*) ';
        ELSE
        T = T || DATA || ') ';
        IF FOUND THEN
        DO;
                                                                /*                          */
                                                                /* NOW BUILD THE PRINT LINE */
                                                                /*                          */
            IF LENGTH(TBUFF) + LENGTH(T) > MAXWIDTH THEN
            DO;
                OUTPUT = TBUFF;
                IF PUNCHTOG THEN
                OUTPUT(2) = TBUFF;
                TBUFF = '  '|| T;
            END; ELSE
            TBUFF = TBUFF || T;
        END;
        IF ¬ EXPRES THEN
                                                                /*                          */
                                                                /* ADVANCE TO NEXT SUBCLASS */
                                                                /*                          */
        CURADR = CURADR - 1;
    END;
    CALL W_CLEAR;
    CALL LINE_FEED(1);
END PRINT_OUTPUT_POOL;
                /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                /*                                              */
                /*                 W_ALL                        */
                /*  OUTPUTS THE VARIOUS TABLES AND STACKS       */
                /*  DEPENDING ON WHICH FLAGS ARE SET.           */
                /*  ROUTINES CALLED:                            */
                /*                 W_EXEC                       */
                /*                 W_ADD                        */
                /*                 W_CONS                       */
                /*                 W_VAL                        */
                /*                 W_CONT                       */
                /*                 W_STATE                      */
                /*                 LINE_FEED                    */
                /*                                              */
                /*<><><><><><><><><><><><><><><><><><><><><><><>*/
W_ALL: PROCEDURE;
    CALL LINE_FEED(2);
    OUTPUT    = ' ***T A B L E   D U M P*** AT LOCATION = '||M_CONTROL;
    IF PUNCHTOG THEN
    OUTPUT(2) = ' ***T A B L E   D U M P*** AT LOCATION = '||M_CONTROL;
    IF ALTST THEN CALL W_STATE(CURBLOCK, ' INPUT POOL FOR ');
    IF ALTCN THEN CALL W_CONS;
    IF ALTAD THEN CALL W_ADD ;
    IF ALTVL THEN CALL W_VAL ;
    IF ALTCT THEN CALL W_CONT;
    IF ALTEX THEN CALL W_EXEC;
END W_ALL;
                /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                /*                                              */
                /*                 DECODE                       */
                /*  FORMATS FOR OUTPUT A FIXED 32 BIT VALUE     */
                /*  OF X WITH L AS THE OUTPUT LINE NUMBER       */
                /*  AND IND AS THE THIRD ENTRY IN THE           */
                /*  FORMATTED STRING.  X IS THE 32 BIT          */
                /*  INTERMEDIATE LANGUAGE WORD FROM CSFCODE     */
                /*  TO FORMAT.                                  */
                /*  ROUTINES CALLED:                            */
                /*                 PAD                          */
                /*                 HEX                          */
                /*                                              */
                /*<><><><><><><><><><><><><><><><><><><><><><><>*/
```

```
DECODE: PROCEDURE(L,X,IND);
     DECLARE (I,J,X,L) FIXED, (T,IND) CHARACTER;
     DECLARE (OFF,TYP ,CON,F) FIXED;
     OFF = SHR(X, 24);
     TYP = SHR(X, 16) & "FF";
     CON = SHR(X, 15) & "1";
     F = X & "7FFF";
     T = ' '|| IND || L;
     T = PAD(T, 6,RIGHT);
     T = T || '   '|| OFF;
     T = PAD(T,12,RIGHT);
     T = T ||'|';
     I = OPRANGE(TYP);
     J = OPRANGE(TYP + 1) - I;
     T = T || SUBSTR(OPCODES, I, J);
     T = PAD(T,24,RIGHT);
     IF CON THEN T = T || '|C|';
     ELSE T = T || '| |';
     T = T || F;
     T = PAD(T,33,RIGHT);
     T = T ||'| ';
     T = T || HEX(X);
     T = T || ' | ';
     IF OPTIM_TYPE(L) ¬= 0 THEN
     DO;
          J = OPTIM_TYPE(L);
          T = T || TYPOPTIM(J);
     END;
     OUTPUT=T;
     IF PUNCHTOG THEN
     OUTPUT(2) = T;
END DECODE;
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                                              */
                  /*                FORMATCODEDUMP                */
                  /* PRINTS THE INCOMING DATA AS:                 */
                  /*                                              */
                  /*  BIT       8       8        1          15    */
                  /*        OFFSET  OPCODE     CONS     ADR(HEX)*/
                  /*                         INDICATOR            */
                  /* ROUTINES CALLED:                             */
                  /*                LINE_FEED                     */
                  /*                DECODE                        */
                  /*                                              */
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
FORMATCODEDUMP: PROCEDURE;
     DECLARE I FIXED;
     CALL LINE_FEED(1);
     OUTPUT =           '   ****  FORMATTED INTERMEDIATE CODE DUMP  ****';
     OUTPUT =                '  LOC OFF     OP CODE     CNS ADR    RAW CODE      OPTIM
IZATION';
     OUTPUT =                '       SET                          /ETC
TYPE';
     OUTPUT = '**********************************************************************
****';
     IF PUNCHTOG THEN
     DO;
          OUTPUT(2) = '  ****  FORMATTED INTERMEDIATE CODE DUMP  ****';
          OUTPUT(2) = '  LOC OFF     OP CODE     CNS ADR     RAW CODE      OPTIMIZATIO
N';
          OUTPUT(2) = '       SET                          /ETC                     TYPE';
          OUTPUT(2) = '**************************************************************
***********';
     END;
     DO I = 0 TO CODELOC;
          CALL DECODE( I, CSFCODE(I), ' ');
     END;
     OUTPUT = '**********************************************************************
****';
     IF PUNCHTOG THEN
     OUTPUT(2)='**********************************************************************
*****';
     CALL LINE_FEED(2);
END FORMATCODEDUMP;
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                                              */
                  /*                  I_CONVERT                   */
                  /* CONVERTS AN INTEGER CHARACTER STRING         */
                  /* REPRESENTATION INTO ITS BINARY EQUIVALENT*/
                  /* RETURNS THE FIXED FORMAT BINARY              */
                  /* REPRESENTATION.                              */
                  /*                                              */
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
I_CONVERT: PROCEDURE (S) FIXED;
     DECLARE S CHARACTER, (I,J,K) FIXED;
     K = 0;
     I = LENGTH(S);
```

```
    DO J = 0 TO I - 1;
        K = K * 10 + BYTE(S, J) - "F0";
    END;
    RETURN K;
END I_CONVERT;
                /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                /*                                             */
                /*                C_CONVERT                    */
                /* PUTS THE INCOMING CONSTANT HEX CHAR         */
                /* VALUES INTO STRING FORMAT AND RETURNS       */
                /* THE HEX STRING.                             */
                /*                                             */
                /*<><><><><><><><><><><><><><><><><><><><><><><>*/
C_CONVERT: PROCEDURE(K,L) CHARACTER;
    DECLARE T CHARACTER, (I, J, K, L) FIXED;
    T = " ";
    T = T || " ";                                   /* FORCE NEW STRING        */
    I = 2;
    DO WHILE I < K;
        J = K - I;
        IF J < 16 THEN
        T = T || SUBSTR(BLANKS, 0, J); ELSE
        T = T || BLANKS;
        I = I + 16;
    END;
    I=ADDR(CSFCODE(L));
    L = SHR(ADDR(T), 2);
    L = COREWORD(L) & "FFFFFF";
    DO J = 0 TO K - 1;
        COREBYTE(L + J) = COREBYTE (I + J);
    END;
    RETURN SUBSTR(T,0,K);
END C_CONVERT;
                /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                /*                                             */
                /*                   DUMP_BLOCKS               */
                /* DUMPS ALL BLOCK HEADERS STORED IN           */
                /* CSF CODE UP TO BLKLOC.                      */
                /* ROUTINES CALLED:                            */
                /*                LINE_FEED                    */
                /*                                             */
                /*<><><><><><><><><><><><><><><><><><><><><><><>*/
DUMP_BLOCKS: PROCEDURE;
    DECLARE (I,J,K,L) FIXED;
    DECLARE BLANKS CHARACTER INITIAL('        ');
    I=CODELOC+1;
    DO WHILE(I < BLKLOC);
        J = CSFCODE(I);
        OUTPUT=BLANKS||'BASIC BLOCK #'||SHR(J,16);
        IF PUNCHTOG THEN
        OUTPUT(2) = BLANKS||'BASIC BLOCK #'||SHR(J,16);
        J=CSFCODE(I+1);
        OUTPUT=BLANKS||BLANKS||'BEGINNING AT '||SHR(J,15)||', ENDING AT '||
            (J&"FFFF");
        IF PUNCHTOG THEN
        OUTPUT(2) = BLANKS||BLANKS||'BEGINNING AT '||SHR(J,16)||', ENDING AT '||
            (J&"FFFF");
        J=CSFCODE(I+2);
        OUTPUT=BLANKS||BLANKS||'BLOCK TRAVERSED '||J||' TIMES';
        IF PUNCHTOG THEN
        OUTPUT(2) = BLANKS||BLANKS||'BLOCK TRAVERSED '||J||' TIMES';
        J = CSFCODE(I + 3);
        OUTPUT = BLANKS || BLANKS || J ||' REFERENCES:';
        IF PUNCHTOG THEN
        OUTPUT(2) = BLANKS || BLANKS || J ||' REFERENCES:';
        I=I+BLKSIZE;
        DO K = 1 TO J;
            L = CSFCODE(I);
            I = I + 1;
            OUTPUT=BLANKS||BLANKS||'    '||L;
            IF PUNCHTOG THEN
            OUTPUT(2) = BLANKS||BLANKS||'    '||L;
        END;
    END;
    CALL LINE_FEED(1);
END DUMP_BLOCKS;
                /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                /*                                             */
                /*                     DUMP_ALL                */
                /* DUMPS ALL TABLES AND STACKS                 */
                /* ROUTINES CALLED:                            */
                /*                W_ALL                        */
                /*                FORMATCODEDUMP               */
                /*                DUMP_BLOCKS                  */
                /*                LINE_FEED                    */
                /*                                             */
                /*<><><><><><><><><><><><><><><><><><><><><><><>*/
DUMP_ALL: PROCEDURE;
```

```
     OUTPUT = ' ***C O M P L E T E   T A B L E   D U M P***';
     OUTPUT = ' DUMP REQUESTED FROM BLOCK NUMBER = '||BLK#||' AT LOCATION = '||
              M_CONTROL;
     IF PUNCHTOG THEN
     DO;
         OUTPUT(2) = ' ***C O M P L E T E   T A B L E   D U M P***';
         OUTPUT(2) = ' DUMP REQUESTED FROM BLOCK NUMBER = ' || BLK# || ' AT LOCAT
ION = '|| M_CONTROL;
     END;
     ALTST = 1;
     ALTEX = 1;
     ALTAD = 1;
     ALTCN = 1;
     ALTVL = 1;
     ALTCT = 1;
     CALL W_ALL;
     CALL FORMATCODEDUMP;
     CALL DUMP_BLOCKS;
     OUTPUT = ' ***E N D   O F   C O M P L E T E   T A B L E   D U M P***';
     IF PUNCHTOG THEN
     OUTPUT(2) =' ***E N D   O F   C O M P L E T E   T A B L E   D U M P***';
     CALL LINE_FEED(1);
END DUMP_ALL;
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                                              */
                  /*                 CERROR                       */
                  /* OUTPUTS COMPILER ERROR NUMBER AND THEN       */
                  /* DUMPS ALL TABLES.                            */
                  /* ROUTINES CALLED:                             */
                  /*              ERROR                           */
                  /*              DUMP_ALL                        */
                  /*                                              */
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
CERROR: PROCEDURE (M);
     DECLARE M CHARACTER;
     CALL ERROR('(COMPILER ERROR) ' || M, 10);
     CALL DUMP_ALL;
END CERROR;
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                                              */
                  /*           OUTPUT_OPTIMIZATION                */
                  /* SEARCHES CSFCODE FOR ANY CODE WORD           */
                  /* WITH BIT 16 SET - INDICATES AN               */
                  /* OPTIMIZATION POSSIBLE - AND OUTPUTS IT.      */
                  /* IF AN OPTIMIZATION IS INDICATED, THE         */
                  /* LOWER 15 BITS OF CODE WORD POINT TO          */
                  /* VTAB FOR THE OPTIMIZATION.  OUTPUT IS IN     */
                  /* THE FOLLOWING FORMAT:                        */
                  /* OPTIMIZATION AT X                            */
                  /*      (A1,.....,AN | B, C)                    */
                  /*               X = LOCATION IN CODE           */
                  /*       A1,...AN = CLASS NUMBERS               */
                  /*                  CORRESPONDING TO EACH       */
                  /*                  OPERAND                     */
                  /*               B = CLASS NUMBER OF AN         */
                  /*                  IDENTICAL EXPRESSION        */
                  /*                  THAT HAS OCCURED EARLIER    */
                  /*               C = IDENTICAL CONSTANT IN      */
                  /*                  CONSTANT TABLE              */
                  /* THE TYPE OF OPTIMIZATION IS THEN OUTPUT      */
                  /* AS:                                          */
                  /*          0 = NONE                            */
                  /*          1 = SIMPLIFICATION                  */
                  /*          2 = CONSTANT PROPAGATION            */
                  /*          3 = COMMON SUBEXPRESSION            */
                  /*              ELIMINATION                     */
                  /*        4 = COMMON SUBEXPRESSION THAT         */
                  /*              LINKS TO A CONSTANT PROPAGATIN*/
                  /* THEN AN OVERVIEW OF THE PASSES PER           */
                  /* BLOCK IS PRINTED ALONG WITH A TOTAL FOR      */
                  /* ALL BLOCKS PROCESSED.                        */
                  /* ROUTINES CALLED:                             */
                  /*              LINE_FEED                       */
                  /*              FORMATCODEDUMP                  */
                  /*              W_STATE                         */
                  /*              PAD                             */
                  /*                                              */
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
OUTPUT_OPTIMIZATION: PROCEDURE;
     DECLARE (I, J, K, D, E) FIXED,
             (T, OPTYP) CHARACTER,
             FOUND BIT(1);
                                                   /*                          */
                                                   /* OUTPUT THE CONSTANT TABLE */
                                                   /* REFINEMENT TABLE          */
                                                   /* AND THE CODE FILE, CLASS  */
                                                   /*                          */
     CALL LINE_FEED(2);
```

```
    OUTPUT    = '                                        FINAL OPTIMIZATION RESULTS';
    OUTPUT    = '____________________________________________________________________
____';
    OUTPUT    = '######################################################################
################';
    OUTPUT    = '--------------------------------------------------------------------
-----------------';
    IF PUNCHTOG THEN
    DO;
        OUTPUT(2) = '                                   FINAL OPTIMIZATION RESULTS'
;
        OUTPUT(2) = '_______________________________________________________________
_________';
        OUTPUT(2) = '##############################################################
###################';
        OUTPUT(2) = '_______________________________________________________________
_________';
    END;
    CALL LINE_FEED(1);
    FOUND = FALSE;
    DO I = #PARMS TO CODELOC;
        J = CSFCODE(I);
        K = SHR(J, 16) & "FF";
        D=OP_TYPE(K);
                                                    /*                             */
                                                    /* IF OPCODE IS NOT A          */
                                                    /* DECLARATION OR A LOAD       */
                                                    /* TEST FOR AN OPTIMIZATION    */
                                                    /*                             */
        IF (K>PASS)&(D¬=LOAD) THEN
        DO;
            D = OP_DEGL(K);
            K = J & "7FFF";
            J = SHR(J,15) & "1";
            J = J & (OPTIM_TYPE(I) ¬= "0");
                                                    /*                             */
                                                    /* IF BIT 16 IS SET, AN        */
                                                    /* OPTIMIZATION TYPE IS        */
                                                    /* PRESENT AND AN ADDRESS FOR*/
                                                    /* THE OPTIMIZATION EXPRESSIN*/
                                                    /* IS PRESENT--WE HAVE AN      */
                                                    /* OPTIMIZATION.               */
                                                    /*                             */
            IF (K¬=0)&J THEN
            DO;
                FOUND = TRUE;
                J = D;
                OUTPUT=' OPTIMIZATION AT '||I;
                IF PUNCHTOG THEN
                OUTPUT(2) = ' OPTIMIZATION AT '||I;
                T='       (';
                DO WHILE (D > 0);
                    D = D - 1;
                    T = T || VTAB(K - D);
                    IF D > 0 THEN T=T||',';
                END;
                T = T || '|';
                D = VTAB(K - J);
                IF D = 0 THEN T = T || '*,';
                ELSE T = T || D || ',';
                D=VTAB(K-J-1);
                IF D = 0 THEN T = T || '*)';
                ELSE T = T || D || ')';
                T =  PAD(T, 30, RIGHT);
                E = OPTIM_TYPE(I);
                OPTYP = TYPOPTIM(E);
                T = T || OPTYP;
                OUTPUT=T;
                IF PUNCHTOG THEN
                OUTPUT(2) = T;
                CALL LINE_FEED(1);
            END;
        END;
            IF OPTIM_TYPE(I) = "0" THEN
            CSFCODE(I) = CSFCODE(I) & "FFFF7FFF";
    END;
    IF ¬FOUND THEN
    DO;
        OUTPUT = ' ***NO OPTIMIZATIONS DETECTED***';
        IF PUNCHTOG THEN
        OUTPUT(2) = ' ***NO OPTIMIZATIONS DETECTED***';
    END;
    CALL FORMATCODEDUMP;
                                                    /*                             */
                                                    /* OUTPUT THE MINIMAL STATE    */
                                                    /* AT EACH BLOCK               */
                                                    /*                             */
    CALL LINE_FEED(2);
```

```
    OUTPUT = '          FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:';
    IF PUNCHTOG THEN
    OUTPUT(2) = '          FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:';
    I = 1;
    DO WHILE I <= BLKCNT;
        CALL LINE_FEED(1);
        J = BLKPNT(I);
        CALL W_STATE(J, ' FINAL POOL FOR ');
        I = I + 1;
    END;
                                                      /*                         */
                                                      /* OUTPUT OVERVIEW OF BLOCKS */
                                                      /* PROCESSED.                */
                                                      /*                         */
    CALL LINE_FEED(2);
    CALL DUMP_BLOCKS;
    OUTPUT = ' BLOCK SUMMARY DATA:';
    IF PUNCHTOG THEN
    OUTPUT(2) = ' BLOCK SUMMARY DATA:';
    CALL LINE_FEED(1);
    OUTPUT = ' BLOCK NUMBER                         NUMBER OF PASSES';
    IF PUNCHTOG THEN
    OUTPUT(2) = ' BLOCK NUMBER                         NUMBER OF PASSES';
    CALL LINE_FEED(1);
    I = 1;
    D = 0;
    DO WHILE I <= BLKCNT;
        T = ' ';
        K =BLKPNT(I);
        J = CSFCODE(K);
        J = SHR(J, 16) & "FFFF";
        T =  PAD(T, 6, RIGHT);
        T = T || J;
        T = PAD(T, 40, RIGHT);
        K = K + 2;
        J = CSFCODE(K);
        D = D + J;
        OUTPUT = T || J ;
        IF PUNCHTOG THEN
        OUTPUT(2) = T || J ;
        I = I + 1;
    END;
    CALL LINE_FEED(2);
    T = ' TOTAL NUMBER OF BLOCKS PROCESSED WAS ' || D;
    OUTPUT = T;
    IF PUNCHTOG THEN
    OUTPUT(2) = T;
    T = ' USING THE ';
    T = T || SELECT_METHOD(SELECT_FLAG);
    T = T || ' BLOCK SELECTION ALGORITHM.';
    OUTPUT = T;
    IF PUNCHTOG THEN
    OUTPUT(2) = T;
END OUTPUT_OPTIMIZATION;
              /*<><><><><><><><><><><><><><><><><><><><><><>*/
              /*                                            */
              /*                  GET_CON                   */
              /* ADVANCES THE CONSTANT POINTER AND THEN     */
              /* CHECKS TO SEE IF IT HAS GONE BEYOND ITS    */
              /* LIMIT.                                     */
              /* ROUTINES CALLED:                           */
              /*                  ERROR                     */
              /*                                            */
              /*<><><><><><><><><><><><><><><><><><><><><><>*/
GET_CON: PROCEDURE;
    CONLOC=CONLOC+1;
    IF CONLOC>MAXCON THEN
    DO;
        CALL ERROR(' CONSTANT TABLE OVERFLOW', 1);
        CONLOC = 1;
    END;
END GET_CON;
              /*<><><><><><><><><><><><><><><><><><><><><><>*/
              /*                                            */
              /*                  ENTERCONS                 */
              /* PUT DATA INTO THE CONSTANT TABLES BY       */
              /* CONLOC.  CONLOC IS FIRST UPDATED AND       */
              /* THEN DATA (S,T) IS PUT INTO NEW CONSTANT   */
              /* ENTRY AS FOLLOWS:                          */
              /*          CONSYM = S - STRING               */
              /*          CONTYPE = T - FIXED               */
              /*          CONINT = 0                        */
              /* ROUTINES CALLED:                           */
              /*                  GET_CON                   */
              /*                                            */
              /*<><><><><><><><><><><><><><><><><><><><><><>*/
ENTERCONS: PROCEDURE(S,T);
    DECLARE S CHARACTER, T FIXED;
```

```
     ALTCN=1;
     CALL GET_CON;
     CONINT(CONLOC)=0;
     CONSYM(CONLOC) = S;
     CONTYPE(CONLOC) = T;
END ENTERCONS;
                    /*<><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                           */
                    /*              ENTERCONI                    */
                    /* PUTS DATA INTO CONSTANT TABLES;           */
                    /*         CONINT = X - FIXED                */
                    /*         CONTYPE = T - FIXED               */
                    /*         CONSYM = X - FIXED                */
                    /* THIS IS BASICALLY THE SAME AS ENTERCONS   */
                    /* EXCEPT A FIXED INTEGER ENTRY FOR SYMBOL   */
                    /* AND THE INTERNAL REPRESENTATION IS SET    */
                    /* TO X.                                     */
                    /* ROUTINES CALLED:                          */
                    /*              GET_CON                      */
                    /*                                           */
                    /*<><><><><><><><><><><><><><><><><><><><><><>*/
ENTERCONI: PROCEDURE (X,T);
     DECLARE (X,T) FIXED;
     ALTCN=1;
     CALL GET_CON;
     CONINT(CONLOC) = X;
     CONTYPE(CONLOC) = T;
     CONSYM(CONLOC) = X;
END ENTERCONI;
                    /*<><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                           */
                    /*              GETVAL#                      */
                    /* RETURNS VALUE NUMBER.  CALLED WITH FORMAL*/
                    /* AND COMPU.                                */
                    /*       COMPU = RETURN A NEW VALUE #        */
                    /*      FORMAL = RETURN 0                    */
                    /*                                           */
                    /*<><><><><><><><><><><><><><><><><><><><><><>*/
GETVAL#: PROCEDURE(I) FIXED;
     DECLARE I FIXED;
     IF I=FORMAL THEN RETURN 0;
     VALUE_NUM_CNTR = VALUE_NUM_CNTR + 1;
     I = VALUE_NUM_CNTR;
     RETURN I;
END GETVAL#;
                    /*<><><><><><><><><><><><><><><><><><><><><><>*/
                    /*                                           */
                    /*           BUILD_CONS_TABLE                */
                    /* BUILDS THE CONSTANT TABLES FROM THE       */
                    /* INPUT DATA IN CSFCODE.                    */
                    /* ROUTINES CALLED:                          */
                    /*            C_CONVERT                      */
                    /*            ENTERCONS                      */
                    /*            GETVAL#                        */
                    /*                                           */
                    /*<><><><><><><><><><><><><><><><><><><><><><>*/
BUILD_CONS_TABLE: PROCEDURE;
     DECLARE (I,J,K,T,L,A) FIXED, Q CHARACTER;
     J=CODELOC+2;
                                              /*                            */
                                              /* PROCESS FROM LAST OF       */
                                              /* CSFCODE FILE TO END OF     */
                                              /* INPUT READ AREA            */
                                              /*                            */
     DO WHILE J <= LASTCON;
          L = CSFCODE(J);
          T = SHR(L, 16) & "FF";
          A = L & "FFFF";
          L = SHR(L, 24);
          Q=C_CONVERT(L,J+1);
                                              /*                            */
                                              /* ENTER CONSTANT INTO TABLE */
                                              /*                            */
          CALL ENTERCONS(Q, T);
          CONADD(CONLOC) = A;
          J=J+SHR(L+7,2);
     END;
                                              /*                            */
                                              /* ASSIGN VALUE #'S TO ALL    */
                                              /* CONSTANTS                  */
                                              /*                            */
     DO I = 1 TO CONLOC;
          CONVAL(I) = GETVAL#(COMPU);
     END;
END BUILD_CONS_TABLE;
```

```
                 /*<><><><><><><><><><><><><><><><><><><><><><>*/
                 /*                                           */
                 /*                READ CODE                  */
                 /* ROUTINE READS THE INPUT FILE INTO         */
                 /* CSFCODE.  A PRESET RECORD SIZE OF 900     */
                 /* WORDS IS ASSUMMED, HOWEVER THE FIRST WORD*/
                 /* OF EACH RECORD READ CONTAINS THE WORD     */
                 /* COUNT OF THE RECORD READ.  THIS VALUE     */
                 /* IS USED TO INSURE THAT ALL THE DATA IS    */
                 /* PROPERLY READ.                            */
                 /* ROUTINES CALLED:                          */
                 /*              ERROR                        */
                 /*              FORMATCODEDUMP               */
                 /*              FILE                         */
                 /*                                           */
                 /*<><><><><><><><><><><><><><><><><><><><><><>*/
READ_CODE: PROCEDURE;
                                                /*                          */
                                                /* READ LENGTH OF INTERMEDIAT*/
                                                /* LANGUAGE.                */
                                                /*                          */
    DECLARE (I,J,K) FIXED;
    CSFCODE(0)=FILE(CSF_FILE,0);
                                                /*                          */
                                                /* SET # OF WORDS READ      */
                                                /*                          */
    CODELOC=CSFCODE(0);
    IF CODELOC > MAXCODE THEN
    DO;
        CALL ERROR (' PROGRAM TOO LARGE TO BE PROCESSED', 4);
        RETURN;
    END;
    I = DISKWORDS;
    J = 0;
                                                /*                          */
                                                /* READ 900 WORD RECORDS    */
                                                /*                          */
    DO WHILE I <= CODELOC;
        J = J + 1;
        CSFCODE(I)=FILE(CSF_FILE,J);
        I=I+DISKWORDS;
    END;
                                                /*                          */
                                                /* INPUT THE CONSTANT COUNT */
                                                /*                          */
    I = CODELOC + 1;
    J = J + 1;
    CSFCODE(I)=FILE(CSF_FILE,J);
    LASTCON = I + CSFCODE(I);
    I = I + DISKWORDS;
                                                /*                          */
                                                /* READ 900 WORD RECORDS    */
                                                /*                          */
    DO WHILE I <= LASTCON;
        J = J + 1;
        I=I+DISKWORDS;
    END;
    IF TRACETOG THEN
    CALL FORMATCODEDUMP;
                                                /*                          */
                                                /* CLEAR THE OPTIMIZATION   */
                                                /* TYPE TABLE FOR THE SIZE  */
                                                /* OF CODE READ             */
                                                /*                          */
    VALUE_NUM_CNTR = CODELOC;
    DO I = 0 TO CODELOC;
         OPTIM_TYPE(I) = "0";
    END;
END READ_CODE;
                 /*<><><><><><><><><><><><><><><><><><><><><><>*/
                 /*                                           */
                 /*                LOOKADD                    */
                 /* SEARCH FOR AN IDENTIFIER IN THE ADDRESS   */
                 /* TABLES                                    */
                 /*    NTFND...RETURN 0                       */
                 /*    FOUND...RETURN THE LOCATION            */
                 /*                                           */
                 /*<><><><><><><><><><><><><><><><><><><><><><>*/
LOOKADD: PROCEDURE (ADR) FIXED;
    DECLARE (ADR, CURADR, FNDADR) FIXED;
    CURADR = ADDLOC;
    FNDADR = 0;
    DO WHILE (CURADR > 0) & (FNDADR = 0);
        IF ADDRESS(CURADR) = ADR THEN FNDADR = CURADR;
        CURADR = CURADR - 1;
    END;
    RETURN FNDADR;
END LOOKADD;
```

```
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                                              */
                  /*                ENTERADD                      */
                  /* ENTERS AN IDENTIFIER INTO THE ADDRESS        */
                  /* TABLES.                                      */
                  /* ROUTINES CALLED:                             */
                  /*                ERROR                         */
                  /*                                              */
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
ENTERADD: PROCEDURE(ADR);
     DECLARE ADR FIXED;
     ADDLOC = ADDLOC + 1;
     ALTAD=1;
     IF ADDLOC > ADDLOCR THEN
     DO;
          CALL ERROR('ADDRESS TABLE OVERFLOW', 1);
          ADDLOC = 1;
     END;
     ADDRESS(ADDLOC) = ADR;
END ENTERADD;
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                                              */
                  /*                LOOKCONS                      */
                  /* LOOKS FOR A CHARACTER STRING AND TYPE        */
                  /* IN THE CONSTANT TABLES.                      */
                  /* RETURNS 0 IF NOT FOUND...                    */
                  /* RETURNS THE LOCATION IF FOUND....            */
                  /*                                              */
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
LOOKCONS: PROCEDURE(SYMSTRG, TYP) FIXED;
     DECLARE SYMSTRG CHARACTER, (FNDADR, TYP, CURADR) FIXED;
     CURADR = CONLOC;
     FNDADR = 0;
     DO WHILE (CURADR > 0) & (FNDADR = 0);
          IF (SYMSTRG = CONSYM(CURADR))& (CONTYPE(CURADR) = TYP) THEN
          FNDADR = CURADR;
          CURADR = CURADR - 1;
     END;
     RETURN FNDADR;
END LOOKCONS;
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                                              */
                  /*                LOOKCONI                      */
                  /* CHECKS FOR A CONSTANT VALUE AND TYPE IN      */
                  /* THE CONSTANT TABLE.                          */
                  /*   TRUE = RETURNS THE LOCATION FOUND          */
                  /*   FALSE = RETURNS 0                          */
                  /*                                              */
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
LOOKCONI: PROCEDURE (VAL#, TYP) FIXED;
     DECLARE (VAL#, TYP, CURADR, FNDADR) FIXED;
     CURADR = CONLOC;
     FNDADR = 0;
     DO WHILE (CURADR > 0) & (FNDADR = 0);
          IF VAL# = CONINT(CURADR) THEN
          IF TYP = CONTYPE(CURADR) THEN
          FNDADR = CURADR;
          CURADR = CURADR - 1;
     END;
     RETURN FNDADR;
END LOOKCONI;
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                 HASHV                        */
                  /*                                              */
                  /*  HASHES USING THE VALUE #'S AND THE          */
                  /* OPERATOR MOD HASHBASE.                       */
                  /*   OP = THE OPERATOR                          */
                  /*  IND = THE INDICATOR FOR THE CURRENT         */
                  /*         EXEC STACK ENTRIES OR VALUE TABLE*/
                  /*         ENTRIES.                             */
                  /*     0 IS THE CURRENT EXEC STACK              */
                  /*     1 ALREADY EXISTS IN THE VALUE TABLE      */
                  /*  HASH IS FORMED:                             */
                  /*  OPA + .... + OPN + OPERAND + F              */
                  /*  HASHBASE IS CURRENTLY SET TO 127.           */
                  /*                                              */
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
HASHV: PROCEDURE (OP, IND) FIXED;
     DECLARE (OP, IND, NUMOP, SUM) FIXED;
     NUMOP = OP_DEGL(OP);
     SUM = OP;
     DO WHILE NUMOP > 0;
          NUMOP = NUMOP - 1;
          DO CASE IND;
               SUM = SUM + EXVAL# (ETOP - NUMOP);
               SUM = SUM + EXNAM# (ETOP - NUMOP);
          END;
     END;
     RETURN (SUM + IND) MOD HASHBASE;
END HASHV;
                  /*<><><><><><><><><><><><><> . . .
```

```
          /*<><><><><><><><><><><><><><><><><><><><><><><>*/
          /*                                              */
          /*              LOOKUPV                         */
          /* SEARCHES FOR AN EXACT VALUE TABLE ENTRY      */
          /* FOR AN OPERATION AND:                        */
          /*     1.  NOT FOUND ... RETURNS 0              */
          /*     2.  FOUND ... RETURNS FWA OF VALUE       */
          /*                   TABLE ENTRY                */
          /* ALL EXPRESSION TABLE ENTRIES FORMED BY       */
          /* THE SAME OPERAND ARE CHAINED IN THE          */
          /* EXPRESSION TABLE AND ALL ARE CHECKED FOR     */
          /* AN EXACT MATCH.                              */
          /*     O = OPERATOR                             */
          /*     F = SWITCH TO DETERMINE HOW TO FORM      */
          /*         HASHCODE                             */
          /*    ST = TOP OF EXPRESSION TABLE              */
          /*         (EFFECTIVELY LIMITS SEARCH)          */
          /* ROUTINES CALLED:                             */
          /*              HASHV                           */
          /*                                              */
          /*<><><><><><><><><><><><><><><><><><><><><><><>*/
LOOKUPV: PROCEDURE (O,F,ST) FIXED;
    DECLARE(O,F, VADR, TSTADR, TSTOP, NUMOPERANDS, OPADR, ST,EXADR, RESADR)
            FIXED,
            (OK, FOUND) BIT(1);
                                                  /*                            */
                                                  /* OPERTR = OPERATOR CODE     */
                                                  /* RESIND = FORMAL/COMPU      */
                                                  /* EADR = EXEC STK POINTER    */
                                                  /* HASHADR = HASHCODE OF ADR  */
                                                  /*           IN VTAB          */
                                                  /*  I = VTAB SUBSCRIPTOR      */
                                                  /*                            */
    OPERTR = O;
    RESIND = F;
    EADR = ETOP;
    VADR = HASHV(O, F);
    HASHADR = VADR;
    TSTADR = 0;
    VADR = VTAB(VADR);
                                                  /*                            */
                                                  /* FIND ALL ENTRIES THAT HASH*/
                                                  /* TO SAME LOCATION.          */
                                                  /*    HASH = OPERANDA +       */
                                                  /*           OPERANDB +       */
                                                  /*           OPERATOR         */
                                                  /*                            */
    DO WHILE VADR > HASHBASE;
        TSTOP = VTAB(VADR);
        TSTOP = TSTOP & "FF";
                                                  /*                            */
                                                  /* CHECK ALL ENTRIES BELOW    */
                                                  /* PREDEFINED LIMIT (ST) THAT*/
                                                  /* HAVE SAME OPERATOR (K = O)*/
                                                  /*                            */
        IF (VADR <= ST) & (TSTOP = O) THEN
        DO;
            NUMOPERANDS = OP_DEGL(O);
            OPADR = VADR + 2;
            OK = TRUE;
                                                  /*                            */
                                                  /* IS OPERATOR COMMUNITIVE    */
                                                  /* VARIABLES  E = EXEC STK    */
                                                  /*                POINTER     */
                                                  /*            N = VTAB POINTR*/
                                                  /*                            */
            IF OP_TYPE(O) = COMM THEN
            DO;
                                                  /*                            */
                                                  /* YES...CHECK EACH OPERAND   */
                                                  /* AGAINST EXPRESSION TABLE   */
                                                  /* ENTRY                      */
                                                  /*                            */
                NUMOPERANDS = NUMOPERANDS - 1;
                EXADR = ETOP - NUMOPERANDS;
                RESADR = OPADR + NUMOPERANDS;
                                                  /*                            */
                                                  /* ACTUAL CHECK OF OPERANDS   */
                                                  /* WHICH CHECKS THE EXPRESSIN*/
                                                  /* ENTRY AGAINST THE INPUT    */
                                                  /* OPERANDS                   */
                                                  /*                            */
                DO WHILE OK & (EXADR <= ETOP);
                    TSTADR = OPADR;
                    FOUND = FALSE;
                    DO WHILE TSTADR <= RESADR;
                        DO CASE F;
                            IF VTAB(TSTADR) = EXVAL#(EXADR) THEN
                            FOUND = TRUE;
```

```
                        IF VTAB(TSTADR) = EXNAM#(EXADR) THEN
                        FOUND = TRUE;
                    END;
                                                     /*                              */
                                                     /* MARK THE EXPRESSION          */
                                                     /* OPERAND IF IDENTICAL SO WE*/
                                                     /* WONT FIND IT AGAIN           */
                                                     /*                              */
                    IF FOUND THEN
                    DO;
                        VTAB(TSTADR) = SHL(1, 15) | VTAB(TSTADR);
                        TSTADR = RESADR;
                    END;
                    TSTADR = TSTADR + 1;
                END;
                                                     /*                              */
                                                     /* ADVANCE TO NEXT OPERAND IN*/
                                                     /* EXEC STACK                   */
                                                     /*                              */
                OK = FOUND;
                EXADR = EXADR + 1;
            END;
                                                     /*                              */
                                                     /* TAKE INDICATOR BIT OFF       */
                                                     /*                              */
            DO OPADR = OPADR TO RESADR;
                VTAB(OPADR) = VTAB(OPADR) & "7FFF";
            END;
        END; ELSE                                    /* NON-COMMUTATIVE OPERATOR     */
                                                     /* OPERANDS MUST BE IN SAME     */
                                                     /* ORDER                        */
        DO WHILE(NUMOPERANDS > 0) & OK;
            NUMOPERANDS = NUMOPERANDS - 1;
            DO CASE F;
                IF VTAB(OPADR) ¬= EXVAL#(ETOP-NUMOPERANDS) THEN OK = TRUE;
                IF VTAB(OPADR) ¬= EXNAM#(ETOP-NUMOPERANDS) THEN OK = TRUE;
            END;
            OPADR = OPADR + 1;
        END;
        IF OK THEN RETURN VADR;
    END;
                                                     /*                              */
                                                     /* ADVANCE TO NEXT EXPRESSION*/
                                                     /* TABLE ENTRY MARKED IN THE */
                                                     /* COLLISION FIELD.             */
                                                     /*                              */
    VADR = VTAB(VADR + 1);
  END;
  RETURN FALSE;
END LOOKUPV;
           /*<><><><><><><><><><><><><><><><><><><><><><><>*/
           /*                                              */
           /*                 ENTERV                       */
           /* BUILDS AN EXPRESSION TABLE ENTRY AS          */
           /* FOLLOWS:                                     */
           /*    X  |  0/1..HASHCD..OPERATOR               */
           /*   +1  |  COLLISION FIELD..0 IF NOT USED      */
           /*   +2  |  VALUE # FOR OPERAND A               */
           /*   +3  |  VALUE # FOR OPERANDB ETC            */
           /*          (POSSIBLY MORE THAN 2 OPERANDS)     */
           /*   +4  |  VALUE # FOR RESULT                  */
           /*   +5  |  POINT TO LAST VALUE TABLE ENTRY     */
           /*          (MASTER CHAIN)                      */
           /* USES FOLLOWING VARIABLE:                     */
           /*          OPERTR   = OPERATOR                 */
           /*          HASHADR  = COLLISION HASH ADDRESS */
           /*          RESIND   = INDICATOR FOR WHERE      */
           /*                      RESULT VALUE # IS FOUND*/
           /*                    0 = EXEC STACK            */
           /*                    1 = IN VALUE TABLE        */
           /*          EADR     = BASE IN EXEC STACK FOR */
           /*                      FIRST OPERAND           */
           /* THE NEW ENTRY IS ALWAYS SET TO BE FIRST      */
           /* ENTRY AWAY FROM THE MASTER HASHADR (LIFO)*/
           /* RETURNS FWA OF NEW EXPRESSION TABLE          */
           /* ENTRY.                                       */
           /* ROUTINES CALLED:                             */
           /*                 ERROR                        */
           /*                                              */
           /*<><><><><><><><><><><><><><><><><><><><><><><>*/
ENTERV: PROCEDURE FIXED;
  DECLARE (I, NUMOP, ADR) FIXED;
  ALTVL=1;
  I = VTOP + 1;
  NUMOP = OP_DEGL(OPERTR);
  VTOP = VTOP + NUMOP + VTSPACE + 3;
  IF VTOP >=VTOPR THEN
  DO;
```

```
        CALL ERROR(' VALUE TABLE OVERFLOW', 1);
        VTOP = HASHBASE;
        VTOPR=MAXVAL+1;
     END;
     VTAB(VTOP) = I - 1;
                                                        /*SET IN INDICATOR, MASTER    */
                                                        /*HASH ADDRESS AND OPERATOR   */
     VTAB(I) = SHL(RESIND, 15) | SHL(HASHADR, 8) | OPERTR;
                                                        /*                            */
                                                        /* SET COLLISION FIELD TO     */
                                                        /* EITHER 0 = NO COLLISION    */
                                                        /*        ¬=0 = COLLISION HASH */
                                                        /*              ADDRESS OF NEXT*/
                                                        /*              VALUE ENTRY    */
                                                        /*                            */
     VTAB(I + 1) = VTAB(HASHADR);
                                                        /*                            */
                                                        /* SET MASTER HASH ADDRESS TO*/
                                                        /* POINT TO LAST VALUE TABLE */
                                                        /* ENTRY IF COLLISION OCCURS */
                                                        /*                            */
     VTAB(HASHADR) = I;
                                                        /*SET IN THE OPERANDS         */
     ADR = I;
     I = I + 2;
     DO WHILE NUMOP > 0;
         NUMOP = NUMOP - 1;
         DO CASE RESIND;
             VTAB(I) = EXVAL#(EADR - NUMOP);
             VTAB(I) = EXNAM#(EADR - NUMOP);
         END;
         I=I+1;
     END;
                                                        /*RETURN FWA OF NEW ENTRY     */
     RETURN ADR;
 END ENTERV;
               /*<><><><><><><><><><><><><><><><><><><><><><><>*/
               /*                                              */
               /*                   REENTERV                   */
               /* ENTERS A VALUE TABLE ENTRY VIA THE           */
               /* COLLISION FIELD OF A PREVIOUS ENTRY          */
               /*     O = OPERAND                              */
               /*     F = OPERATION LOCATION (SEE HASHV        */
               /*         ENTRY)                               */
               /* RETURNS FWA OF NEW VALUE TABLE ENTRY         */
               /* ROUTINES CALLED:                             */
               /*                 HASHV                        */
               /*                                              */
               /*<><><><><><><><><><><><><><><><><><><><><><><>*/
 REENTERV: PROCEDURE(O,F) FIXED;
     DECLARE (O,F) FIXED;
     OPERTR = O;
     RESIND = F;
     EADR = ETOP;
     HASHADR = HASHV(O, F);
     RETURN ENTERV;
 END REENTERV;
               /*<><><><><><><><><><><><><><><><><><><><><><><>*/
               /*                                              */
               /*                   REMOVEV                    */
               /* REMOVES VALUE TABLE ENTRIES ABOVE TOP.       */
               /* LINKS THE COLLISION FIELD OF REMOVED         */
               /* ENTRY TO MAINTAIN CHAIN.                     */
               /*                                              */
               /*<><><><><><><><><><><><><><><><><><><><><><><>*/
 REMOVEV:PROCEDURE (TOP);
     DECLARE (TOP, HSHADR) FIXED;
     ALTVL=1;
     DO WHILE VTOP > TOP;
         VTOP = VTAB (VTOP);
         HSHADR = VTAB(VTOP + 1);
         HSHADR = SHR(HSHADR, 8) & "7F";
         VTAB(HSHADR) = VTAB(VTOP + 2);
     END;
 END REMOVEV;
               /*<><><><><><><><><><><><><><><><><><><><><><><>*/
               /*                                              */
               /*                    GET_ST                    */
               /* ADVANCE THE STATE POINTER AND CHECK FOR      */
               /* OVERFLOW.                                    */
               /* ROUTINES CALLED:                             */
               /*                 ERROR                        */
               /*                                              */
               /*<><><><><><><><><><><><><><><><><><><><><><><>*/
 GET_ST: PROCEDURE;
     STLOC=STLOC+1;
     IF STLOC > MAXSTATE THEN
     DO;
```

```
            CALL ERROR(' STATE TABLE OVERFLOW', 1);
            STLOC=1;
        END;
END GET_ST;
                  /*<><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                                            */
                  /*             GET_ADDR                       */
                  /* DECREMENT ADDLOCR AND TEST FOR OVERLAP     */
                  /* WITH ADDLOC                                */
                  /* ROUTINES CALLED:                           */
                  /*             ERROR                          */
                  /*                                            */
                  /*<><><><><><><><><><><><><><><><><><><><><><>*/
GET_ADDR: PROCEDURE;
        ADDLOCR=ADDLOCR-1;
        IF ADDLOCR < = ADDLOC THEN
        DO;
            CALL ERROR(' STATE ADDRESS TABLE OVERFLOW', 1);
            ADDLOCR=MAXADD;
        END;
END GET_ADDR;
                  /*<><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                                            */
                  /*                  GET_EXECR                 */
                  /* ADVANCES EXEC STACK TOP POINTER AND        */
                  /* CHECKS FOR UNDERFLOW.                      */
                  /* ROUTINES CALLED:                           */
                  /*             ERROR                          */
                  /*                                            */
                  /*<><><><><><><><><><><><><><><><><><><><><><>*/
GET_EXECR: PROCEDURE;
        ETOPR=ETOPR-1;
        IF ETOPR <=ETOP THEN
        DO;
            CALL ERROR(' EXEC STACK OVERFLOW (REVERSE) ', 1);
            ETOPR=MAXEXEC;
        END;
END GET_EXECR;
                  /*<><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                                            */
                  /*             GET_EXEC                       */
                  /* ADVANCE THE EXEC STACK POINTER AND         */
                  /* CHECK FOR OVERFOLW                         */
                  /* ROUTINES CALLED:                           */
                  /*             ERROR                          */
                  /*                                            */
                  /*<><><><><><><><><><><><><><><><><><><><><><>*/
GET_EXEC: PROCEDURE;
        ETOP=ETOP+1;
        IF ETOP >= ETOPR THEN
        DO;
            CALL ERROR (' EXEC STACK OVERFLOW (FORWARD) ', 1);
            ETOP=1;
        END;
END GET_EXEC;
                  /*<><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                                            */
                  /*             GET_VTOPR                      */
                  /* DECREMENT VTOPR AND THEN CHECK TO SEE IF   */
                  /* IT HAS DECREASED TO VTOP.  ERROR IS SENT   */
                  /* IF ABOVE OCCURS.                           */
                  /* ROUTINES CALLED:                           */
                  /*             ERROR                          */
                  /*                                            */
                  /*<><><><><><><><><><><><><><><><><><><><><><>*/
GET_VTOPR: PROCEDURE;
        VTOPR=VTOPR-1;
        IF VTOPR < VTOP THEN
        DO;
            CALL ERROR(' VALUE TABLE OVERFLOW (FROM TOP) ', 1);
            VTOPR = MAXVAL;
            VTOP = HASHBASE;
        END;
END GET_VTOPR;
                  /*<><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                                            */
                  /*                   PERMUTE                  */
                  /* PERMUTES OPERANDS STARTING AT A CORE       */
                  /* ADDRESS OF ADR IN A TABLE STRUCTURED       */
                  /* WITH #BYTE BYTES PER ENTRY.  REUTRNS A     */
                  /* TRUE CONDITION IF A PERMUTATION POSSIBLE.*/
                  /* WHEN CALLED FOR THE FIRST TIME, FIRSTCALL*/
                  /* MUST EQUAL TRUE IN ORDER TO PRESET         */
                  /* POINTERS.                                  */
                  /* ADOPTED FROM CACM ALGORITHM 115.           */
                  /*                                            */
                  /*<><><><><><><><><><><><><><><><><><><><><><>*/
PERMUTE: PROCEDURE(NUMOP, ADR, #BYTE, METHOD) BIT(1);
```

```
    DECLARE (NUMOP, ADR, #BYTE, LOOP, PTA, PTB, I) FIXED,
            DATA BIT(8),
            (PERMFLG, FOUND, METHOD) BIT(1);
    PERMFLG = FALSE;
    IF FIRSTCALL THEN
    DO;
        FIRSTCALL = FALSE;
        BOPA = 1;
        BOPB = 1;
    END;
    BOPB = BOPB + 1;
    IF BOPB > NUMOP THEN
    DO;
        BOPA = BOPA + 1;
        BOPB = BOPA + 1;
        IF BOPA < NUMOP THEN
        PERMFLG = TRUE;
    END; ELSE
    PERMFLG = TRUE;
    LOOP = #BYTE - 1;
    IF PERMFLG THEN
    DO;
        PTA = (BOPA - 1) * #BYTE;
        PTB = (BOPB - 1) * #BYTE;
        IF METHOD THEN
        DO;
            PTA = ADR + PTA;
            PTB = ADR + PTB;
        END; ELSE
        DO;
            PTA = ADR - PTA;
            PTB = ADR - PTB;
        END;
        DO I = 0 TO LOOP;
            DATA = COREBYTE(PTA + I);
            COREBYTE(PTA + I) = COREBYTE(PTB + I);
            COREBYTE(PTB + I) = DATA;
        END;
    END;
    RETURN PERMFLG;
END PERMUTE;
                /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                /*                                             */
                /*              INSRT_SUC_BLOCK                */
                /* INSERTS TO THE TOP OF THE BLOCK TO BE       */
                /* PROCESSED LIST.  THE NUMBER OF EXITS ARE    */
                /* ADVANCED AND TOGGLES ARE SET FOR BLOCK      */
                /* IN QUESTION.                                */
                /*     BIT 31     30-16       15-0             */
                /*              PREDECESSOR  BLK FWA IN CODE   */
                /*                 BLOCK#                      */
                /*     1 = BLOCK DOES NOT NEED TO BE           */
                /*         PROCESSED                           */
                /*     0 = BLOCK NEEDS TO BE PROCESSED         */
                /*                                             */
                /* ROUTINES CALLED:                            */
                /*              ERROR                          */
                /*                                             */
                /*<><><><><><><><><><><><><><><><><><><><><><><>*/
INSRT_SUC_BLOCK: PROCEDURE(FWA);
    DECLARE (FWA, TEMA, I) FIXED;
    ALTCT=1;
    CTOP=CTOP+1;
    IF CTOP > MAXCONT THEN
    DO;
        CALL ERROR('CONTROL TABLE OVERFLOW', 1);
        CTOP=1;
    END;
    CONTROL(CTOP) = SHL(BLK#, 16) | FWA;
    #XITS=#XITS+1;
    TEMA = 0;
                                              /*                          */
                                              /* BUILD THE TOGGLE WORD FOR */
                                              /* THE BLOCK AND SAVE IT     */
                                              /*                          */
    DO I = 0 TO #TOGS;
        TEMA = TEMA | SHL(TOGGLES(I), I);
    END;
    SAV_TOG(CTOP) = TEMA;
END INSRT_SUC_BLOCK;
```

```
                /*<><><><><><><><><><><><><><><><><><><><><><>*/
                /*                                           */
                /*              PROPAGATE                    */
                /* PERFORMS THE TOP OPERATION OF THE STACK   */
                /* OR PRESETS THE BINARY VALUE OF A CONSTANT*/
                /* ACCORDING TO ITS TYPE.                    */
                /* ROUTINES WITHIN PROCEDURE:                */
                /*              SET_IJ                       */
                /*              ENTERI                       */
                /*              ENTERB                       */
                /* ROUTINES CALLED:                          */
                /*              I_CONVERT                    */
                /*              SET_IJ                       */
                /*              ENTERI                       */
                /*              ERROR                        */
                /*              ENTERB                       */
                /*                                           */
                /*<><><><><><><><><><><><><><><><><><><><><><>*/
PROPAGATE: PROCEDURE(CASE#,SUBCASE#) FIXED;
    DECLARE (CASE#,SUBCASE#,RESULT,I,J,K,L) FIXED;
                /*<><><><><><><><><><><><><><><><><><><><><><>*/
                /*                                           */
                /*              SET_IJ                       */
                /* EXTRACTS THE TOP 2 CONSTANT VALUES FROM   */
                /* THE STACK.                                */
                /*   I = TOP                                 */
                /*   J = TOP - 1                             */
                /*                                           */
                /*<><><><><><><><><><><><><><><><><><><><><><>*/
    SET_IJ: PROCEDURE;
        I = EXADD(ETOP);
        J = EXADD(ETOP - 1);
        I = CONINT(I);
        J = CONINT(J);
    END SET_IJ;
                /*<><><><><><><><><><><><><><><><><><><><><><>*/
                /*                                           */
                /*              ENTERI                       */
                /* CHECKS FOR AN INTEGER VALUE ALREADY IN    */
                /* THE CONSTANT TABLE.  IF NOT FOUND A NEW   */
                /* ENTRY FOR THE CONSTANT IS MADE IN THE     */
                /* CONSTANT TABLES.                          */
                /* ROUTINES CALLED:                          */
                /*              LOOKCONI                     */
                /*              ENTERCONI                    */
                /*              GETVAL#                      */
                /*                                           */
                /*<><><><><><><><><><><><><><><><><><><><><><>*/
    ENTERI: PROCEDURE;
        RESULT=LOOKCONI(K,INT);
        IF RESULT=0 THEN
        DO;
            CALL ENTERCONI(K, INT);
            RESULT = CONLOC;
            CONVAL(RESULT)=GETVAL#(COMPU);
        END;
    END ENTERI;
                /*<><><><><><><><><><><><><><><><><><><><><><>*/
                /*                                           */
                /*              ENTERB                       */
                /* CHECKS FOR A BOOLEAN VALUE ALREADY IN THE*/
                /* CONSTANT TABLES.  IF NOT FOUND THE NEW    */
                /* VALUE IS PUT INTO THE CONSTANT TABLES.    */
                /* ROUTINES CALLED:                          */
                /*              ENTERCONI                    */
                /*              GETVAL#                      */
                /*                                           */
                /*<><><><><><><><><><><><><><><><><><><><><><>*/
    ENTERB: PROCEDURE;
        RESULT=LOOKCONI(K,BOOL);
        IF RESULT=0 THEN
        DO;
            CALL ENTERCONI(K, BOOL);
            RESULT = CONLOC;
            CONVAL(RESULT)=GETVAL#(COMPU);
            IF K THEN CONSYM(RESULT) = 'T';
            ELSE CONSYM(RESULT) = 'F';
        END;
    END ENTERB;
    RESULT=0;
                                                  /*                            */
                                                  /* RESULT WILL BE SET TO      */
                                                  /* ADDRESS OF RESULTING       */
                                                  /* COMPUTATION IN CONSTANT    */
                                                  /* TABLE                      */
                                                  /*                            */
                                                  /*                            */
                                                  /* BEGINNING OF PROPAGATE     */
                                                  /*                            */
 DO CASE CASE#;
                                                  /* CASE 0: NULL               */
        ;
                                                  /* CASE 1: DEFAULT => (TYPE) */
        ;
                                                  /* CASE 2: LOC => (TYPE)      */
        ;
```

```
                                                  /* CASE 3: ENT => (TYPE)       */
    ;
                                                  /* CASE 4: XIT => (TYPE)       */
    ;
                                                  /* CASE 5: INT => (TYPE)       */
                                                  /* CONVERT FROM EXTERNAL TO    */
                                                  /* INTERNAL FORM               */
DO;
    I = EXADD(ETOP);
    CONINT(I)=I_CONVERT(CONSYM(I));
END;
                                                  /* CASE 6: REAL => (TYPE)      */
    ;
                                                  /* CASE 7: BOOL => (TYPE)      */
DO;
    I = EXADD(ETOP);
    IF CONSYM(I) = 'T' THEN CONINT(I) = 1;
    ELSE CONINT(I) = 0;
END;
                                                  /* CASE 8: IARRAY => (TYPE)    */
    ;
                                                  /* CASE 9: RARRAY => (TYPE)    */
    ;
                                                  /* CASE 10: REFER => (OPCODE)*/
    ;
                                                  /* CASE 11: TOGGLE=> (OPCODE)*/
    ;
                                                  /* CASE 12: PASS => (OPCODE) */
    ;
                                                  /* CASE 13: TRU REAL =>        */
                                                  /*          INT(OPCODE)        */
    ;
                                                  /* CASE 14: RND REAL =>        */
                                                  /*          INT (OPCODE)       */
    ;
                                                  /* CASE 15: FLT INT =>         */
                                                  /*          REAL (OPCODE)      */
    ;
                                                  /* CASE 16: ADD INT INT =>     */
                                                  /*          INT (OPCODE)       */
DO;
    CALL SET_IJ;
    K = J + I;
    CALL ENTERI;
END;
                                                  /* CASE 17: RADD REAL REAL =>*/
                                                  /*          REAL (OPCODE)      */
    ;
                                                  /* CASE 18: SUB INT INT =>     */
                                                  /*          INT (OPCODE)       */
DO;
    CALL SET_IJ;
    K = J - I;
    CALL ENTERI;
END;
                                                  /* CASE 19: RSUB REAL REAL =>*/
                                                  /*          REAL (OPCODE)      */
    ;
                                                  /* CASE 20: MUL INT INT =>     */
                                                  /*          INT (OPCODE)       */
DO;
    CALL SET_IJ;
    K = J * I;
    CALL ENTERI;
END;
                                                  /* CASE 21: RMUL REAL REAL =>*/
                                                  /*          REAL (OPCODE)      */
    ;
                                                  /* CASE 22: DIV INT INT =>     */
                                                  /*          INT (OPCODE)       */
DO;
    CALL SET_IJ;
    IF I = 0 THEN
    CALL ERROR(' ATTEMPT TO DIVIDE BY ZERO', 3);
    ELSE
    DO;
        K = I / J;
        CALL ENTERI;
    END;
END;
                                                  /* CASE 23: RDIV REAL REAL =>*/
                                                  /*          REAL (OPCODEE      */
    ;
                                                  /* CASE 24: EXP INT INT =>     */
                                                  /*          INT (OPCODE)       */
    ;
                                                  /* CASE 25: RIXP REAL INT => */
                                                  /*          REAL (OPCODE)      */
```

```
        ;
                                        /* CASE 26: IRXP INT REAL => */
                                        /*          REAL (OPCODE)    */
        ;
                                        /* CASE 27: RRXP REAL REAL =>*/
                                        /*          REAL (OPCODE)    */
        ;
                                        /* CASE 28: LSS DEFAULT      */
                                        /*          DEFAULT => BOOL  */
                                        /*        . (OPCODEE         */
    DO;
        CALL SET_IJ;
        K = J < I;
        CALL ENTERB;
    END;
                                        /* CASE 29: LEQ DEFAULT      */
                                        /* DEFAULT => BOOL (OPCODE)  */
    DO;
        CALL SET_IJ;
        K = J <= I;
        CALL ENTERB;
    END;
                                        /* CASE 30: EQL DEFAULT      */
                                        /* DEFAULT => BOOL (OPCODE)  */
    DO;
        CALL SET_IJ;
        K = J = I;
        CALL ENTERB;
    END;
                                        /* CASE 31: NEQ DEFAULT      */
                                        /* DEFAULT => BOOL (OPCODE)  */
    DO;
        CALL SET_IJ;
        K = J ¬= I;
        CALL ENTERB;
    END;
                                        /* CASE 32: GEQ DEFAULT      */
                                        /* DEFAULT => BOOL (OPCODE)  */
    DO;
        CALL SET_IJ;
        K = J >= I;
        CALL ENTERB;
    END;
                                        /* CASE 33: GTR DEFAULT      */
                                        /* DEFAULT => BOOL (OPCODE)  */
    DO;
        CALL SET_IJ;
        K = J > I;
        CALL ENTERB;
    END;
                                        /* CASE 34: INEG INT => INT  */
                                        /* (OPCODE)                  */
    DO;
        I = EXADD(ETOP);
        I = CONINT(I);
        K = -I;
        CALL ENTERB;
    END;
                                        /* CASE 35: RNEG REAL => REAL*/
                                        /* (OPCODE)                  */
        ;
                                        /* CASE 36: NOT BOOL => BOOL */
                                        /* (OPCODE)                  */
    DO;
        I = EXADD(ETOP);
        I = CONINT(I);
        K =¬ I;
        CALL ENTERB;
    END;
                                        /* CASE 37: AND BOOL BOOL  > */
                                        /* BOOL (OPCODE)             */
    DO;
        CALL SET_IJ;
        K = J & I;
        CALL ENTERB;
    END;
                                        /* CASE 38: BOR BOOL BOOL  > */
                                        /* BOOL (OPCODE)             */
    DO;
        CALL SET_IJ;
        K = J | I;
        CALL ENTERB;
    END;
                                        /* CASE 39: LOD LOC =>       */
                                        /* DEFAULT (OPCODE)          */
        ;
                                        /* CASE 40: STO LOC DEFAULT  */
                                        /* => DEFAULT (OPCODE)       */
```

```
          ;
                                                      /* CASE 41: STO LOC DEFAULT  */
                                                      /* => (OPCODE)               */
          ;
                                                      /* CASE 42: DEL DEFAULT =>   */
                                                      /* (OPCODE)                  */
          ;
                                                      /* CASE 43: DUP => DEFAULT   */
                                                      /* (OPCODE)                  */
          ;
                                                      /* CASE 44: XCH => (OPCODE)  */
          ;
                                                      /* CASE 45: HLT => (OPCODE)  */
          ;
                                                      /* CASE 46: BRS XIT =>       */
                                                      /* (OPCODE)                  */
          ;
                                                      /* CASE 47: BSC BOOL XIT =>  */
                                                      /* (OPCODE)                  */
                                                      /*                           */
      DO;
                                                      /* CONTROL STACK             */
                                                      /*                           */
          I = EXADD(ETOP);
          I = CONINT(I);
          IF I THEN                                   /* IS TOP ENTRY ON EXEC STACK*/
                                                      /* A CONSTANT                */
          DO;
                                                      /*                           */
                                                      /* YES...STRIP IT OFF... IE.,*/
                                                      /* SET UP TO TAKE TRUE BRANCH*/
                                                      /*                           */
              SAV_TOG(CTOP - 1) = SAV_TOG(CTOP);
              CONTROL(CTOP-1)=CONTROL(CTOP);
          END;
                                                      /*                           */
                                                      /* NO...SET UP TO TAKE THE   */
                                                      /* FALSE BRANCH              */
                                                      /*                           */
          #XITS=#XITS-1;
          CTOP=CTOP-1;
      END;
                                                      /* CASE 48: NOP => (OPCODE)  */
          ;
                                                      /* CASE 49: PRO XIT =>       */
                                                      /* (OPCODE)                  */
          ;
                                                      /* CASE 50: RTN => (OPCODE)  */
          ;
                                                      /* CASE 51: GET INT => LOC   */
                                                      /* (OPCODE)                  */
          ;
                                                      /* CASE 52: RET LOC =>       */
                                                      /* (OPCODE)                  */
          ;
                                                      /* CASE 53: RRD => REAL      */
                                                      /* (OPCODE)                  */
          ;
                                                      /* CASE 54: IRD => INT       */
                                                      /* (OPCODE)                  */
          ;
                                                      /* CASE 55: BRD => BOOL      */
                                                      /* (OPCODE)                  */
          ;
                                                      /* CASE 56: WRV DEFAULT =>   */
                                                      /* (OPCODE)                  */
          ;
                                                      /* CASE 57: DMP => (OPCODE)  */
          ;
                                                      /* CASE 58: TAB INT =>       */
                                                      /* (OPCODE)                  */
          ;
                                                      /* CASE 59: SUP DEFAULT =>   */
                                                      /* (OPCODE)                  */
          ;
   END;
      RETURN RESULT;
END PROPAGATE; /*
```

```
                   /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                   /*                                              */
                   /*               BASIC_BLOCK                    */
                   /* BLOCK ANALYZER DONE BY OPERATOR TYPE.        */
                   /* ENTRY IS MADE WITH M_CONTROL SET .           */
                   /* PROCESSING USES M_CONTROL AS THE POINTER     */
                   /* TO CSFCODE.  CODE WORDS ARE PROCESSED        */
                   /* UNTIL A BRANCH OPERATOR OR AN ENTRY POINT    */
                   /* IS ENCOUNTERED.  THE INPUT POOL FOR THE      */
                   /* BLOCK TO BE PROCESSED IS PRESET BEFORE       */
                   /* ENTRY AND UPON EXIT THE OUTPUT POOL IS       */
                   /* PRESENT.  FOUR TYPES OF OPTIMIZATIONS        */
                   /* ARE CHECKED FOR:                             */
                   /*      1 = COMMON SUBEXPRESSIONS               */
                   /*      2 = CONSTANT PROPAGATIONS               */
                   /*      3 = SIMPLIFICATIONS                     */
                   /*      4 = COMMON SUBEXPRESSIONS THAT LINK     */
                   /*          TO CONSTANT PROPAGATION             */
                   /*                                              */
                   /* ROUTINES WITHIN PROCEDURE:                   */
                   /*              PROP_CONS                       */
                   /*              SIMPLIFY                        */
                   /*              CONSTANT_CLASS                  */
                   /* ROUTINES CALLED:                             */
                   /*              DECODE                          */
                   /*              CERROR                          */
                   /*              LOOKADD                         */
                   /*              ENTERADD                        */
                   /*              GETVAL#                         */
                   /*              INSRT_SUC_BLOCK                 */
                   /*              GET_EXEC                        */
                   /*              LOOKUPV                         */
                   /*              CONSTANT_CLASS                  */
                   /*              REENTERV                        */
                   /*              SIMPLIFY                        */
                   /*              SET_SUBCLASS                    */
                   /*              GET_VTOPR                       */
                   /*              ENTERV                          */
                   /*              ERROR                           */
                   /*              PROPAGATE                       */
                   /*                                              */
                   /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                   /*                                              */
BASIC_BLOCK: PROCEDURE;
     DECLARE (CODE, TYP, OP, A, V, N, I, J, FIRST, LAST, FORM, SUBCASE,
              VPOINT, VT, NT, EA, NUMOP, COUNT) FIXED,
             (DEGL, DEGR, DEFTYPE, RESTYPE) BIT(8),
             (RETAIN, C, SUBEXP) BIT(1);
                   /*                                              */
                   /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                   /*                                              */
                   /*               SEARCH_OPTIM_TABLE             */
                   /* SEARCHES FOR AN EXACT EXPRESSION MATCH       */
                   /* IN THE OPTIMIZATION AREA.  IF FOUND          */
                   /* RETURNS THE LOCATION OF THE OPTIMIZATION     */
                   /* THIS ROUTINE IS USED TO IDENTIFY             */
                   /* IDENTICAL OPTIMIZATIONS PREVIOUSLY           */
                   /* COMPUTED.                                    */
                   /* ROUTINES CALLED:                             */
                   /*              PERMUTE                         */
                   /*              ERROR                           */
                   /*                                              */
                   /*<><><><><><><><><><><><><><><><><><><><><><><>*/
SEARCH_OPTIM_TABLE: PROCEDURE(ADR) FIXED;
     DECLARE (ADR, NUMOP, I, J, OPSFND, SRCHADR, FOUND, INTAB,
             COMOP, BASEADR, NUM, PERM, OPLFT) FIXED;
     OPLFT = TRUE;
     FIRSTCALL = TRUE;
     COMOP = OP_TYPE(OP) = COMM;
     FOUND = FALSE;
     INTAB = FALSE;
     I = OPTOP - 1;
     DO WHILE ( I > 0) & (¬FOUND);
          SRCHADR = OPTIM_ADR_LIST(I);
          IF SRCHADR = ADR THEN
          INTAB = TRUE;
          ELSE
          DO WHILE OPLFT;
               NUM, NUMOP = DEGL;
               J = 0;
               OPSFND = 0;
               DO WHILE NUMOP > 0;
                    IF VTAB(SRCHADR - J) = VTAB(ADR - J) THEN
                    OPSFND = OPSFND + 1;
                    NUMOP = NUMOP - 1;
                    J = J + 1;
               END;
               IF OPSFND = DEGL THEN
               DO;
                    IF VTAB(SRCHADR - DEGL - 1) = VTAB(ADR - DEGL - 1) THEN
                    DO;
                         IF VTAB(SRCHADR - DEGL) < VTAB(ADR - DEGL) THEN
                         DO;
                              OPLFT = FALSE;
                              FOUND = VTAB(SRCHADR - DEGL);
                         END;
                    END;
```

```
                END;
                IF ¬FOUND THEN
                DO;
                     IF COMOP THEN
                     DO;
                          BASEADR = ADDR(VTAB(ADR));
                          PERM = PERMUTE(NUM, BASEADR, 2, FALSE);
                          IF ¬PERM THEN
                          OPLFT = FALSE;
                     END;
                     ELSE
                     OPLFT = FALSE;
                END;
           END;
           OPLFT = TRUE;
           FIRSTCALL = TRUE;
           I = I - 1;
      END;
      IF ¬INTAB THEN
      DO;
           OPTIM_ADR_LIST(OPTOP) = ADR;
           OPTOP = OPTOP + 1;
           IF OPTOP > MAX_OPTIMIZATIONS THEN
           DO;
                CALL ERROR('MAXIMUM OPTIMIZATIONS EXCEEDED', 7);
                OPTOP = OPTOP - 1;
           END;
      END;
      RETURN FOUND;
      END SEARCH_OPTIM_TABLE;
                  /*<><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                                              */
                  /*                SET_OPTIM_LOC                 */
                  /* SEARCHES THE PAST OPTIMIZATION ADDRESS       */
                  /* FOR A EXACT DUPLICATE TO SET IN THE          */
                  /* ADDRESS INTO THE OPTIMIZATION AREA.          */
                  /* IF NO MATCH IS FOUND THE NEW ENTRY IS        */
                  /* ENTERED FOR FUTURE SEARCHES.                 */
                  /* ROUTINES CALLED;                             */
                  /*              SEARCH_OPTIM_TABLE              */
                  /*                                              */
                  /*<><><><><><><><><><><><><><><><><><><><><><>*/
SET_OPTIM_LOC: PROCEDURE (ADR);
      DECLARE (ADR, SAME_OPTIM) FIXED;
      IF ADR = 0 THEN
      RETURN;
      SAME_OPTIM = SEARCH_OPTIM_TABLE(ADR);
      IF SAME_OPTIM ¬= 0 THEN
      VTAB(ADR - DEGL) = SAME_OPTIM;
END SET_OPTIM_LOC;
                  /*<><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                                              */
                  /*                PROP_CONS                     */
                  /* INSURES THAT ALL OPERANDS ARE CONSTANTS.     */
                  /* IF SO, THE OPERATION IS PERFORMED.           */
                  /* RETURNS            0 IF NOT ALL CONSTANTS    */
                  /*                  ¬=0 THE ADDRESS OF ANSWER   */
                  /*                      IN CONSTANT TABLE       */
                  /* ROUTINES CALLED:                             */
                  /*                PROPAGATE                     */
                  /*                                              */
                  /*<><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                                              */
      PROP_CONS: PROCEDURE FIXED;
           DECLARE NUMOP FIXED, CONFLG BIT(1);
           CONFLG = TRUE;
           NUMOP = DEGL;
                                                   /*                             */
                                                   /* CHECK ALL OPERANDS FOR AN   */
                                                   /* OPERATOR FOR A CONSTANT     */
                                                   /* BIT IN THE EXEC STACK       */
                                                   /*                             */
           DO WHILE (NUMOP > 0) & CONFLG;
                NUMOP = NUMOP - 1;
                CONFLG = EXCON(ETOP - NUMOP);
           END;
           IF CONFLG THEN                          /* IF ALL OPERANDS ARE         */
                                                   /* CONSTANTS CALL PROPAGATE    */
                                                   /* TO FORM TO ANSWER TO THE    */
                                                   /* DESIRED UPERATION           */
           RETURN PROPAGATE(OP, SUBCASE);
           RETURN 0;
      END PROP_CONS;
                  /*                                            */
```

```
                 /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                 /*                                             */
                 /*              SIMPLIFY                       */
                 /* USED IN THE CASE OF A SIMPLIFICATION        */
                 /* OPERATOR.  A SIMPLIFICATION IS SEARCHED     */
                 /* FOR SUCH AS:                                */
                 /*              0 + A = A                      */
                 /*              1 * A = A                      */
                 /*              0 * A = 0       ETC.           */
                 /* ROUTINES CALLED:                            */
                 /*              PROP_CONS                      */
                 /*              ERROR                          */
                 /*              LOOKCONS                       */
                 /*              GET_EXECR                      */
                 /*              PERMUTE                        */
                 /*              ENTERCONS                      */
                 /*              PROPAGATE                      */
                 /*              GETVAL#                        */
                 /*                                             */
                 /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                 /*                                             */
SIMPLIFY: PROCEDURE;
      DECLARE (POSIT, RLPNT, TYP, SAVRLPNT, OPFORM, NUMOP, EXPNT, IND,
            CONSADR, STPNT, RESPNT, VALADR, FIRSTPERM, OPADR) FIXED,
            (EXTOP, DATA, ID#, RESTYP) BIT(16),
            (MATCH, COMOP) BIT(1);
      EA = PROP_CONS;                          /* CHECK FOR ALL CONSTANT     */
                                               /* OPERANDS.  IF THIS IS MET,*/
                                               /* AN OBVIOUS CONSTANT       */
                                               /* PROPAGATION IS PRESENT AND*/
                                               /* CAN BE SIMPLIFIED.        */
                                               /*                           */
      IF EA¬=0 THEN
      DO;
                                               /*                           */
                                               /* YES...SET VAL# IN V.      */
                                               /*                           */
           V = CONVAL(EA);
                                               /*                           */
                                               /* IF COMMON SUBEXPRESSION   */
                                               /* ALREADY DETECTED, INDICATE*/
                                               /* LINK AND SIMPLY MARK AS   */
                                               /* CONSTANT PROPAGATION.     */
                                               /*                           */
           IF OPTIM_TYPE(M_CONTROL) = "3" THEN
           OPTIM_TYPE(M_CONTROL) = "4";
           ELSE
           OPTIM_TYPE(M_CONTROL) = "2";
           RETURN;
      END;
                                               /*                           */
                                               /* NO...ANSWER MUST BE VTAB  */
                                               /* ENTRY.                    */
                                               /*                           */
      POSIT = 0;
      EXTOP = ETOPR;
                                               /*                           */
                                               /* MAKE SURE STATE TABLE WILL*/
                                               /* NOT OVERFLOW              */
                                               /*                           */
      IF (STLOC+DEGL) > MAXSTATE THEN
      DO;
           CALL ERROR('STATE TABLE OVERFLOW (2)', 1);
           RETURN;
      END;
                                               /*                           */
                                               /* SET UP POINTERS TO        */
                                               /* SIMPLIFICATION RULES.     */
                                               /*                           */
      RLPNT = SIMP_INDEX(OP);
      TYP = OP_INFO(RLPNT);
      COMOP = OP_TYPE(OP) = COMM;
      FIRSTCALL = TRUE;
      MATCH = FALSE;
                                               /*                           */
                                               /* PROCESS WHILE             */
                                               /* SIMPLIFICATION RULE IS    */
                                               /* PRESENT.                  */
                                               /*                           */
      DO WHILE (RLPNT ¬= 0) & (¬MATCH);
           SAVRLPNT = RLPNT;
                                               /*                           */
                                               /* SET UP TO COMPARE RULE    */
                                               /* FORM AGAINST EXPRESSION   */
                                               /* ON EXEC STACK.            */
                                               /*                           */
           OPFORM = FORM;
           NUMOP = 0;
           MATCH = TRUE;
           ETOPR = EXTOP;
           POSIT = 0;
                                               /*                           */
```

```
                                                          /* CHECK TYPE OF ALL OPERANDS*/
                                                          /* IN EXEC STACK FOR A MATCH */
                                                          /* AGAINST THE SIMPLIFICATION*/
                                                          /* RULE                      */
                                                          /*                         */
        DO WHILE (NUMOP < DEGL) & MATCH;
           EXPNT = ETOP - NUMOP;
           RLPNT = RLPNT + 1;
           DATA = OP_INFO(RLPNT);
           IND = DATA & "1";
           ID# = SHR(DATA, 1);
           IF IND THEN                                    /* IS OPERAND RULE FOR AN    */
                                                          /*IDENTITY CONSTANT          */
           DO;
               MATCH = FALSE;                             /* YES...                    */
               IND = OP_INFO(OPFORM);                     /* IS EXEC STACK OPERAND A   */
                                                          /* CONSTANT AND OF PROPER    */
                                                          /* TYPE FOR EXPRESSION.      */
               IF EXCON(EXPNT) & (EXTYPE(EXPNT) = IND) THEN
               DO;
                                                          /*                         */
                                                          /* YES...ADDRESS OF THE      */
                                                          /* IDENTITY IN CONSTANT TABLE*/
                                                          /*                         */
                   CONSADR = LOOKCONS(OP_STR(ID#), IND);
                                                          /*                         */
                                                          /* MATCH IF EXEC CONSTANT IS */
                                                          /* EQUAL TO THE IDENTITY IN  */
                                                          /* THE RULE.                 */
                                                          /*                         */
                   MATCH = (CONSADR ¬= 0) & (CONSADR = EXADD(EXPNT));
               END;
                                                          /*                         */
                                                          /* NOT A CONSTANT, IE...     */
                                                          /* MUST BE VARIABLE OPERAND. */
           END; ELSE                                      /* ALWAYS TRUE AT LEAST ONCE */
                                                          /* ASKING HAS EXEC STACK BEEN*/
                                                          /* SAVED.                    */
           IF POSIT < ID# THEN
           DO;
                                                          /*                         */
                                                          /* NO...SAVE EXEC STACK      */
                                                          /* ELEMENT AT TOP OF EXEC STK*/
                                                          /*                         */
               POSIT = ID#;
               CALL GET_EXECR;
               EXCON(ETOPR) = EXCON(EXPNT);
               EXADD(ETOPR) = EXADD(EXPNT);
               EXVAL#(ETOPR) = EXVAL#(EXPNT);
               STATES(STLOC + ID#) = ETOPR;
           END; ELSE
           DO;
               MATCH = FALSE;
               STPNT = STATES(STLOC + ID#);
                                                          /*                         */
                                                          /*IS THE SAVED VARIABLE EQUAL*/
                                                          /*TO THE RESULT VARIABLE     */
                                                          /*                         */
               IF EXVAL#(STPNT) = EXVAL#(EXPNT) THEN
               MATCH = TRUE;
               ELSE
                                                          /*                         */
                                                          /* ARE EITHER CURRENT OR     */
                                                          /* SAVED OPERANDS CONSTANTS  */
                                                          /*                         */
               IF EXCON(EXPNT) | EXCON(STPNT) THEN
               DO;
                                                          /*                         */
                                                          /* YES...THE SAME CONSTANT   */
                                                          /*                         */
                   IF EXADD(STPNT) ¬= EXADD(EXPNT) THEN
                   DO;
                                                          /*                         */
                                                          /* NO...CLEAR THE CONSTANT   */
                                                          /* ENTRY IN THE SAVED ENTRY  */
                                                          /* IE..VARIABLE OPERAND      */
                                                          /*                         */
                       EXCON(STPNT) = 0;
                       EXADD(STPNT) = 0;
                   END;
               END;
           END;
                                                          /*                         */
                                                          /* ADVANCE COUNTER AND FORMAL*/
                                                          /* TYPE TO SEARCH FOR THEN   */
                                                          /* BACK FOR NEXT OPERAND     */
                                                          /*                         */
           NUMOP = NUMOP + 1;
           OPFORM = OPFORM + 1;
```

```
               OPFORM = OPFORM + 1;
                                                         /*                            */
                                                         /* END OF (NUMOP < DEGL) &    */
                                                         /* MATCH.                     */
                                                         /*                            */
           END;
           CONSADR = RLPNT;
                                                         /*                            */
                                                         /* WAS A MATCH FOUND          */
                                                         /*                            */

           IF MATCH THEN
                                                         /*                            */
                                                         /* YES...CLEAR SIMPLIFICATION*/
                                                         /* POINTER.                   */
                                                         /*                            */

           RLPNT = 0; ELSE
           DO;
                                                         /*                            */
                                                         /* NO...IS OPERATOR           */
                                                         /* COMMUNITIVE                */
                                                         /*                            */
               IF COMOP THEN
               DO;
                                                         /*                            */
                                                         /*                            */
                                                         /* YES...LETS TRY A           */
                                                         /* PERMUTATION.               */
                   OPADR = ADDR(OP_INFO(SAVRLPNT + 1));
                   FIRSTPERM = PERMUTE(DEGL, OPADR, 2, TRUE);
                   IF FIRSTPERM THEN                     /* WAS PERMUTATION POSS       */
                   RLPNT = SAVRLPNT;                     /* YES...SET FOR SAME RULE    */
                   ELSE
                   DO;                                   /* NO...ADVANCE TO NEXT RULE */
                       FIRSTCALL = TRUE;
                       RLPNT = TYP;
                   END;
               END;
                                                         /*                            */
                                                         /* OPERATOR IS NOT COMMUNITIV*/
                                                         /*                            */

               ELSE
               DO;
                   RLPNT = TYP;
                   FIRSTCALL = TRUE;
               END;
                                                         /*                            */
                                                         /* IF NEW RULE SELECTED,      */
                                                         /* RESET NEXT RULE POINTER.   */
                                                         /*                            */
               IF TYP = RLPNT THEN TYP = OP_INFO(RLPNT);
           END;
                                                         /*                            */
                                                         /* BACK TO TEST FOR NEXT RULE*/
                                                         /* OR TEST A PERMUTATION FOR */
                                                         /* A COMMUNITIVE OPERATOR     */
                                                         /* AGAINST SAME RULE.         */
                                                         /*                            */
       END;                                              /* END OF RLPNT ¬= 0          */
       RLPNT = CONSADR;
       IF MATCH THEN                                     /* WAS A SIMPLIFICATION RULE */
                                                         /* MATCHED.                   */
       DO;
           OPTIM_TYPE(M_CONTROL) = "1";
                                                         /*                            */
                                                         /* YES...CHECK THE RESULTANT */
                                                         /* TYPE, ETC.                 */
                                                         /*                            */
           RLPNT = RLPNT + 1;
           RESTYP = OP_INFO(RLPNT);
           RESPNT = RESTYPE;
           IND = RESTYP & "1";
           RESTYP = SHR(RESTYP, 1);                      /* SHOULD THE RESULT BE AN    */
                                                         /* IDENTITY                   */
           IF IND THEN
           DO;
                                                         /*                            */
                                                         /* YES...IS THE IDENTITY IN  */
                                                         /* THE CONSTANT TABLES        */
                                                         /*                            */
               CONSADR = LOOKCONS(OP_STR(RESTYP), RESPNT);
               IF CONSADR = 0 THEN
               DO;
                                                         /*                            */
                                                         /* NO...PUT IDENTITY IN       */
                                                         /* CONSTANT TABLE.            */
                                                         /*                            */
                   CALL ENTERCONS(OP_STR(RESTYP), RESPNT);
                   CONSADR = CONLOC;
```

```
                                                        /*                          */
                                                        /* PUT IDENTITY VALUE ON TOP */
                                                        /* OF EXEC STACK.            */
                                                        /*                          */
                ETOP = ETOP + 1;
                EXADD(ETOP) = CONSADR;
                CALL PROPAGATE(RESPNT, 0);
                ETOP = ETOP - 1;
                CONVAL(CONSADR) = GETVAL#(COMPU);
            END;
                                                        /*                          */
                                                        /* SET VALUE # BY CONSTANT   */
                                                        /* TABLE.                    */
                                                        /*                          */
            V = CONVAL(CONSADR);
            EA = CONSADR;
        END;
        ELSE
                                                        /* RESULT IS NOT A CONSTANT  */
                                                        /* IDENTITY.                 */
        DO;
            VALADR = STATES(STLOC + RESTYP);
            V = EXVAL#(VALADR);
            EA = EXADD(VALADR);
        END;
    END; ELSE                                           /* NO MATCH, IE., NO         */
                                                        /* SIMPLIFICATION DETECTED.  */
    V, EA = 0;
    ETOPR = EXTOP;
END SIMPLIFY;
        /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
        /*                                                */
        /*              CONSTANT_CLASS                    */
        /* THIS ROUTINE IS ENTERED IF AN EXPRESSION       */
        /* MATCHES A PREVIOUS CLASS.  THE CODEWORD        */
        /* IS MARKED AS OPTIMIZED AND A VALUE NUMBER*/
        /* IS OBTAINED.                                   */
        /*                                                */
        /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
CONSTANT_CLASS: PROCEDURE;
    V = GETVAL#(COMPU);
    VTAB(A - DEGL) = M_CONTROL;
    A = A|SHL (1, 15);
END CONSTANT_CLASS;
                                                        /*                          */
                                                        /* ACTUAL BEGINNING OF BASIC */
                                                        /* BLOCK.                    */
                                                        /* SAVE BEGINNING ADDRESS    */
                                                        /* TABLE POINTER AND CLEAR   */
                                                        /* NUMBER OF EXITS           */
                                                        /*                          */
TADDLOC=ADDLOC;
#XITS=0;
                                                        /*                          */
                                                        /* MASTER CONTROL LOOP -     */
                                                        /* PROCESS UNTIL ENT, BRNCH  */
                                                        /* OPERATOR OR END OF CODE IS*/
                                                        /* ENCOUNTERED               */
                                                        /*                          */
DO WHILE M_CONTROL < CODELOC;
    M_CONTROL = M_CONTROL + 1;
                                                        /*                          */
                                                        /* SET VARIABLES:            */
                                                        /*   CODE = RAW CODE         */
                                                        /* OP,TYP = OP CODE          */
                                                        /*      C = CONSTANT BIT     */
                                                        /*      A = ADDRESS          */
                                                        /*                          */
    CODE = CSFCODE(M_CONTROL);
    IF TRACETOG THEN
    CALL DECODE(M_CONTROL, CODE, '8+');
    OP, TYP = SHR(CODE, 16) & "FF";
    OPERTR = OP;
    C = SHR(CODE, 15) & "1";
    A = CODE & "7FFF";
    ALTEX = 1;
                                                        /*                          */
                                                        /* TOGGLES OPERATOR - SET    */
                                                        /* TOGGLES BY A              */
                                                        /*                          */
    IF OP = TOGGLE THEN
    DO;
        J = SHR(A, 8);
        I = (A & "FF") - 1;
        IF (I <= #TOGS) & (I >= 0) THEN
        TOGGLES(I) = J;
    END; ELSE
    IF TYP <= NTYPES THEN                               /* TAG OR DATA DECLARATION   */
```

```
                                                  /*        0 = NULL              */
                                                  /*        1 = DEFAULT           */
                                                  /*        2 = LOC               */
                                                  /*        3 = ENT               */
                                                  /*        4 = XIT               */
                                                  /*        5 = INT               */
                                                  /*        6 = REAL              */
                                                  /*        7 = BOOL              */
                                                  /*        8 = IARRAY            */
                                                  /*        9 = RARRAY            */
DO;
    N = 0;
    IF TYP <= XIT THEN                            /* SPECIAL CASES                */
    DO CASE TYP;                                  /* NULL,DEFAULT,LOC,ENT,XIT     */
        CALL CERROR('TYPE NULL ENCOUNTERED');
        CALL CERROR('TYPE DEFAULT ENCOUNTERED');
                                                  /*                              */
                                                  /*            LOC               */
                                                  /* LOC TYPE CONSTANT            */
                                                  /*                              */
                                                  /*                              */
                                                  /* IS THIS A CONSTANT           */
                                                  /*                              */
        IF C THEN
                                                  /*                              */
                                                  /* YES...                       */
                                                  /*                              */
        V = 0;
        ELSE
        DO;                                       /* NO...IDENTIFIER SHOULD BE    */
                                                  /* IN ADR TABLE                 */
            IF A = 0 THEN
            CALL CERROR('NULL LOCATION ');
                                                  /*                              */
                                                  /* GO FIND IT                   */
                                                  /*                              */
            I = LOOKADD(A);
            IF I = 0 THEN                         /* IDENTIFIER NOT PREVIOUSLY    */
                                                  /* DEFINED                      */
            DO;
                                                  /*                              */
                                                  /* GET NEW ADDRESS FOR THIS     */
                                                  /* IDENTIFIER                   */
                                                  /*                              */
                CALL ENTERADD(A);
                I = ADDLOC;
                ADDTYPE(I) = NULL;
                ADDVAL(I) = 0;
                ADDCON(I) = 0;
                V, ADDNAM(I) = GETVAL#(FORMAL);
            END;
            ELSE V = ADDNAM(I);
        END;
                                                  /*                              */
                                                  /*            ENT               */
                                                  /* ENT TYPE CONSTANT.. ENTER    */
                                                  /* SET NEW BLOCK NUMBER ON      */
                                                  /* THE CONTROL TABLE FOR        */
                                                  /* FUTURE PROCESSING AND EXIT   */
                                                  /* BACK TO CONTROL FLOW.        */
                                                  /*                              */
        DO;
            CALL INSRT_SUC_BLOCK(M_CONTROL);
            RETURN;
        END;
                                                  /*                              */
                                                  /*            XIT               */
                                                  /* XIT TYPE CONSTANT..SET NEW   */
                                                  /* BLOCK NUMBER ON CONTROL      */
                                                  /* TABLE FOR FUTURE PROCESSIN   */
                                                  /* WITH ENTRY ADDRESS A         */
                                                  /*                              */
        CALL INSRT_SUC_BLOCK(A);
    END; ELSE
                                                  /*                              */
                                                  /* REFERENCE IS TO A STANDARD   */
                                                  /* DECLARATION - BEGIN          */
                                                  /* EXAMINATION FOR EXEC STACK   */
                                                  /* SET UP.                      */
                                                  /*                              */
    IF A = 0 THEN                                 /* REFERENCE TO INDETERMINATE   */
                                                  /* TYPE - I.E., NO IDENTIFIER   */
                                                  /* PRESENT                      */
    DO;
        C = 0;
        V = GETVAL#(COMPU);
        N = GETVAL#(FORMAL);
    END ; ELSE
```

```
                                                      /*                            */
                                                      /* REFERENCE TO DEFINED       */
                                                      /* IDENTIFIER                 */
          DO;
                                                      /*                            */
              C = 1;
              V = CONVAL(A);
              N = 0;
          END;
          IF TYP ¬= XIT THEN                          /* MAKE AN ENTRY INTO THE     */
                                                      /* INTO THE EXEC STACK.  IF   */
                                                      /* OPERATOR IS NOT XIT.       */
          DO;
              CALL GET_EXEC;
              I = ETOP;
              EXNAM#(I) = N;
              EXTYPE(I) = TYP;
              EXCON(I) = C;
              EXADD(I) = A;
              EXVAL#(I) = V;
          END;
      END;
                                                      /* IF TYPE > XIT, DO          */
      ELSE                                            /* OPERATION CODE DETECTED    */
                                                      /* IF THIS POINT REACHED      */
                                                      /*                            */
      DO;                                             /* SETUP NUMBER OF OPERANDS   */
                                                      /* AND NUMBER OF RESULTS      */
                                                      /* BY OPERATOR TYPE.          */
          DEGL = OP_DEGL(OP);
          DEGR = OP_DEGR(OP);
                                                      /*                            */
                                                      /* RETAIN USED TO INDICATE    */
                                                      /* IF TOP OF THE EXEC STACK   */
                                                      /* IS TO BE SAVED UPON        */
                                                      /* COMPLETION OF INSTRUCTION  */
                                                      /*                            */
          RETAIN = 0;
                                                      /*                            */
                                                      /* MASTER CASE ON OPERATOR    */
                                                      /* BASED ON THE OPTYPE TABLE:*/
                                                      /*     0 = SIMPLIFICATION     */
                                                      /*         OPERATOR           */
                                                      /*     1 = UNUSED             */
                                                      /*     2 = LOAD               */
                                                      /*     3 = STORE AND DELETE   */
                                                      /*         TOP OF EXEC STACK  */
                                                      /*     4 = STORE AND RETAIN   */
                                                      /*         TOP OF EXEC STACK  */
                                                      /*     5 = DELETE             */
                                                      /*     6 = DUPLICATE          */
                                                      /*     7 = CONVERSION         */
                                                      /*     8 = COMMUNITIVE        */
                                                      /*     9 = EXCHANGE           */
                                                      /*    10 = CONDITIONAL JUMP   */
                                                      /*    11 = UNCONDITIONAL JUMP */
                                                      /*    12 = GOSUB              */
                                                      /*    13 = RETSUB             */
                                                      /*                            */
          DO CASE OP_TYPE(OP);
                                                      /*                            */
                                                      /* SIMPLIFICATION OPERATOR    */
                                                      /* ALSO USED BY:              */
                                                      /*    A..CONV(7)              */
                                                      /*    B..COMM(8)              */
SIMPLE_OP:
              DO;                                     /* CHECK FORM OF EACH OPERAND*/
                                                      /*                            */
                                                      /* NUMBER OF OPERANDS PROPER  */
                                                      /* AND RESULT TYPE CORRECT    */
                                                      /*                            */
                  IF ETOP < DEGL THEN
                  DO;
                                                      /*                            */
                                                      /* NO, THEN EXIT              */
                                                      /*                            */
                      CALL CERROR('NOT ENOUGH OPERANDS');
                      RETURN;
                  END;
                  FORM = 0;
                  SUBCASE = -1;
                  RESTYPE = DEFAULT;
                                                      /*                            */
                                                      /* SET POINTER TO OPERAND     */
                                                      /* TYPES IN THE OP_INFO TABLE*/
                                                      /*                            */
                  FIRST = OP_INDEX(OP);
                  LAST = OP_INDEX(OP + 1);
```

```
                                                   /*                              */
                                                   /* LOOK FOR PROPER NUMBER OF    */
                                                   /* OPERANDS, TYPE OF OPERANDS   */
                                                   /* AND TYPE OF RESULT.          */
                                                   /*                              */
DO WHILE (FIRST < LAST) & (FORM = 0);
    SUBCASE = SUBCASE + 1;
    DEFTYPE = 0;
    J = ETOP;
    FORM = FIRST;
    I = FIRST + DEGL;
                                                   /*                              */
                                                   /* CHECK TYPE OF ALL OPERANDS   */
                                                   /* BY OPERATOR                  */
                                                   /*                              */
    DO WHILE (FORM¬=0) & (FIRST < I);
        V = OP_INFO(FIRST);
        N = EXTYPE(J);
        IF V = DEFAULT THEN                        /* TYPE NOT IMPORTANT FOR       */
                                                   /* OPERATOR                     */
        DO;
            IF DEFTYPE = 0 THEN
            DEFTYPE = N; ELSE
            DO;
                IF DEFTYPE ¬= N THEN
                FORM = 0;
            END;
        END; ELSE
        DO;                                        /* CHECK FOR FORM OF RESULT     */
                                                   /* AGAINST THE OPERAND.         */
            IF ¬ ((V = N) | (N = NULL)) THEN
            FORM = 0;
        END;
                                                   /*                              */
                                                   /* ADVANCE TO NEXT OPERAND      */
                                                   /*                              */
        FIRST = FIRST + 1;
        J = J - 1;
    END;
                                                   /*                              */
                                                   /* SET IN NUMBER OF RESULTS     */
                                                   /* FOR OPERATOR                 */
                                                   /*                              */
    FIRST = I + DEGR;
END;
IF FORM = 0 THEN                                   /* FORM = 0 SAYS OPERATORS OK   */
                                                   /* SO SET IT TO POINT TO THE    */
                                                   /* PROPER TYPES IN OP_INFO      */
                                                   /* FOR THE OPERATOR.            */
FORM = OP_INDEX(OP);
IF DEGR > 0 THEN                                   /* NOW LOOK AT RESULTANT TYPE   */
                                                   /* SET IT ACCORDING TO OP_INF   */
                                                   /* OR IF NOT IMPORTANT BY THE   */
                                                   /* OPERAND TYPE                 */
DO;
    RESTYPE = OP_INFO(FORM + DEGL + DEGR - 1);
    IF RESTYPE = DEFAULT THEN
    RESTYPE = DEFTYPE;
END;
                                                   /*                              */
                                                   /* END OF CHECK FOR THE         */
                                                   /* CORRECT FORM                 */
                                                   /*                              */
EA = 0;
J, V, N = 0;
C = SHR(CODE, 15) & "1";
                                                   /*                              */
                                                   /* HAS THE OPERATION  ALREADY   */
                                                   /* BEEN DEFINED                 */
                                                   /*                              */
IF TYP ¬= PASS THEN
DO;
    IF A ¬= 0 THEN
    DO;
        NUMOP = DEGL;
        COUNT = 0;
        J = 0;
                                                   /*                              */
                                                   /* YES...CHECK EACH OPERAND     */
                                                   /* TO SEE IF IT HAS BEEN        */
                                                   /* CHANGED SINCE THE PREVIOUS   */
                                                   /* CALCULATION WAS PERFORMED    */
                                                   /*                              */
        DO WHILE NUMOP > 0;
            NUMOP = NUMOP - 1;
                                                   /*                              */
                                                   /* GET OPERAND VALUE #          */
                                                   /* I.E., THE VALUE ASSIGNED     */
```

```
                                  /* TO THE IDENTIFIER              */
                                  /*                                */
    V = EXVAL#(ETOP - NUMOP);
                                  /*                                */
                                  /* COMPARE THE NEW VALUE#         */
                                  /* AGAINST THE OLD EXPRESSION*/
                                  /*                                */
    N = A - J;
    IF VTAB(N) = V THEN
    COUNT = COUNT + 1;
    ELSE
                                  /*                                */
                                  /* SET IN NEW VALUE # FOR         */
                                  /* OPERAND IF CHANGED             */
                                  /*                                */
    VTAB(N) = V;
    J = J + 1;
END;
                                  /*                                */
                                  /* IF AN OPERAND HAS CHANGED      */
                                  /* AND THE OPTIM WAS A COM        */
                                  /* SUB ELLIM THEN RESET IT TO
                                  /* BE NOT OPTIMIZED.
                                  /*                                */
J = COUNT = DEGL;
SUBEXP = (OPTIM_TYPE(M_CONTROL) = "3") |
         (OPTIM_TYPE(M_CONTROL) = "4");
IF (¬ J) & SUBEXP THEN
OPTIM_TYPE(M_CONTROL) = "0";
I, V = 0;
N = A & "7FFF";
                                  /*                                */
                                  /*  SEE IF AN IDENTICAL           */
                                  /*  EXPRESSION EXISTS IN THE      */
                                  /*  VALUE TABLE                   */
                                  /*                                */
I = LOOKUPV(OP, COMPU, VTOP);
                                  /*                                */
IF I > 0 THEN                     /* YES...SET IN THE NEW           */
DO;
                                  /* EXPRESSION RESULT INTO         */
                                  /* THE OLD EXPRESSION ENTRY       */
                                  /*AND MARK AS COMM SUB ELIM       */
                                  /*                                */
    OPTIM_TYPE(M_CONTROL) = "3";
    A = A | SHL(I, 15);
    VPOINT = I + DEGL + 2;
    V, VTAB(N - DEGL - 1) = VTAB(VPOINT);
END; ELSE
DO;
                                  /*                                */
                                  /* NO..HAS THE EXP ALREADY        */
                                  /* BEEN PROCESSED AND NO          */
                                  /* OPERANDS HAVE CHANGED          */
                                  /*                                */
    IF C & J THEN
                                  /*                                */
                                  /* GET THE OLD RESULT VAL#        */
                                  /*                                */
    V = VTAB(N - DEGL - 1);
    ELSE
                                  /*                                */
                                  /* NO..GET A NEW VALUE #          */
                                  /* AND MARK AS PROCESSED          */
                                  /*                                */
    CALL CONSTANT_CLASS;
                                  /*                                */
                                  /* CHAIN NEW VALUE TABLE          */
                                  /* ENTRY TO SAME CLASS AND        */
                                  /* SET IN RESULT VALUE #          */
                                  /*                                */
    I = REENTERV(OP, COMPU);
    VPOINT = I + DEGL + 2;
    VTAB(VPOINT) = V;
END;
I = V;
V = 0;
                                  /*                                */
                                  /* IS PROPAGATION REQUESTED       */
                                  /*                                */
IF PROPTOG THEN
                                  /*                                */
                                  /* YES...TRY TO PERFORM A         */
                                  /* SIMPLIFICATION                 */
                                  /*                                */
CALL SIMPLIFY;
                                  /*                                */
                                  /* IF THE OPTIM IS COM SUB        */
```

```
                          /* EXP AND NO SIMPLIFICATION */
                          /* WAS DETECTED...BYPASS     */
                          /* CLEARING.                 */
                          /*                           */
                          /*                           */
                          /* IF NEW VALUE # ASSIGNED,  */
                          /* SET IT INTO VTAB          */
                          /*                           */
    IF (V = 0) & (OPTIM_TYPE(M_CONTROL) = "3") THEN
    V = I;
    ELSE
    DO;
        IF V = 0 THEN
        DO;
            OPTIM_TYPE(M_CONTROL) = "0";
            V = I;
        END;
    END;
    VTAB(N - DEGL - 1) = V;
    VTAB(VPOINT) = V;
    SUBEXP = (OPTIM_TYPE(M_CONTROL) = "2") |
             (OPTIM_TYPE(M_CONTROL) = "4");
    IF SUBEXP THEN
    ADDCON(VPOINT) = EA;
    I, N = 0;
END; ELSE
                          /*                           */
                          /* NO ADDRESS AVAILABLE.     */
                          /* CHECK FOR FIRST PASS THIS */
                          /* BLOCK                     */
                          /*                           */
IF PASS# = 1 THEN
DO;
                          /*                           */
                          /* YES...CHECK TO SEE IF AN  */
                          /* OPTIMIZATION IS POSSIBLE. */
                          /* BUILD A VTAB ENTRY FOR    */
                          /* THIS EXPRESSION           */
                          /*                           */
    NUMOP = DEGL;
    COUNT = 0;
    A = VTOPR - 1;
                          /*                           */
                          /* SET TO TEST ALL OPERANDS  */
                          /*                           */
    DO WHILE NUMOP > 0;
        N = DEGL - NUMOP;
        NUMOP = NUMOP - 1;
        NT = EXVAL#(ETOP - N);
                          /*                           */
                          /* CHECK IF VALUE # PRESENT  */
                          /* FOR THE OPERANDS ON THE   */
                          /* EXEC STACK.  IE., A       */
                          /* USEFUL DATA IN OPERAND    */
                          /* CHECK.                    */
                          /*                           */
        IF NT ¬= 0 THEN
        DO;
                          /*                           */
                          /* YES...SET INTO TOP OF     */
                          /* VALUE TABLE.  IE., SAVED  */
                          /* OPTIMIZATION INFORMATION  */
                          /* AREA.                     */
                          /*                           */
            COUNT = COUNT + 1;
            CALL GET_VTOPR;
            VTAB(VTOPR) = NT;
        END;
    END;
    IF COUNT ¬= DEGL THEN
    DO;
        VTOPR = A + 1 + DEGL;
        A = 0;
        I = 0;
    END; ELSE
                          /*                           */
                          /* DID ALL OPERANDS HAVE     */
                          /* A VALUE #...INDICATES ALL */
                          /* OPERANDS ARE USEFUL - NOW */
                          /* CHECK FOR A COMMON        */
                          /* SUBEXPRESSION ELIMINATION.*/
                          /*                           */
    DO;
                          /*                           */
                          /* YES...SEE IF A VALUE TABLE*/
                          /* ENTRY EXISTS THAT IS      */
                          /* IDENTICAL                 */
                          /*                           */
```

```
         CALL GET_VTOPR;
         I = LOOKUPV(OP, COMPU, VTOP);
                                 /*                            */
                                 /* YES...IDENTICAL ENTRY      */
         IF I > 0 THEN
         DO;
                                 /*                            */
                                 /* EXISTS - GET THE VALUE #   */
                                 /* OF ITS RESULT AND MARK     */
                                 /* THIS INSTRUCTION AS A      */
                                 /* COMMON SUBEXPRESSION       */
                                 /* ELIMINATION.               */
                                 /*                            */
             VPOINT = I + DEGL + 2;
             A = A | SHL(1, 15);
             V = VTAB(VPOINT);
             OPTIM_TYPE(M_CONTROL) = "3";
         END; ELSE
         DO;
                                 /*                            */
                                 /* NO..GET A NEW VALUE# , SET*/
                                 /* OPTIM BIT AND BUILD A NEW */
                                 /* EXPRESSION ENTRY.          */
                                 /*                            */
             RESIND = COMPU;
             HASHADR = HASHV(OPERTR, COMPU);
             EADR = ETOP;
             V = GETVAL#(COMPU);
             A = A | SHL(1, 15);
             I = ENTERV;
             VPOINT = I + DEGL + 2;
             VTAB(VPOINT) = V;
         END;
                                 /*                            */
                                 /* SET LOCATION INTO TOP OF   */
                                 /* VTAB.                      */
                                 /*                            */
         VTAB(VTOPR) = M_CONTROL;
         I = V;
         CALL GET_VTOPR;
         V = 0;
                                 /*                            */
                                 /* CHECK FOR PROPAGATION      */
                                 /* REQUESTED                  */
                                 /*                            */
         IF PROPTOG THEN
                                 /*                            */
                                 /* YES...TRY TO PERFORM A     */
                                 /* SIMPLIFICATION             */
                                 /*                            */
         CALL SIMPLIFY;
                                 /*                            */
                                 /* SAVE THE RESULT VALUE#     */
                                 /*                            */
         IF (V = 0) & (OPTIM_TYPE(M_CONTROL) = "3") THEN
         V = I;
         ELSE
         DO;
             IF V = 0 THEN
             DO;
                 OPTIM_TYPE(M_CONTROL) = "0";
                 V = I;
             END;
         END;
         VTAB(VPOINT) = V;
         VTAB(VTOPR) = V;
         SUBEXP = (OPTIM_TYPE(M_CONTROL) = "2") |
                  (OPTIM_TYPE(M_CONTROL) = "4");
         IF SUBEXP THEN
         ADDCON(VPOINT) = EA;
     END;
END; ELSE                        /* PASS# IS NOT EQUAL TO 1    */
DO;
    A = 0;
    I = 0;
    V = 0;
END;
N = 0;                           /* SET IN THE ADDRESS OF VTAB*/
                                 /* CONTAINING        OPA      */
                                 /*                   OPB      */
                                 /*                   OPTIM ADR */
                                 /*                   RESULT    */
                                 /* SET A INTO CODE FILE. THIS*/
                                 /* IS THE POINT OF MARKING    */
                                 /* OPTIMIZATION FOR LATER     */
                                 /* DETECTION.                 */
CSFCODE(M_CONTROL) = (CODE & "FFFF8000") | A;
N = A & "7FFF";
```

```
                        IF OPTIM_TYPE(M_CONTROL) ¬= "1" THEN
                        CALL SET_OPTIM_LOC(N);
                        ELSE
                        DO;
                            IF N ¬= 0 THEN
                            DO;
                                N = N - DEGL;
                                VTAB(N) = M_CONTROL;
                            END;
                        END;
                        N = 0;
                                                 /*                          */
                                                 /* NOW REDUCE THE EXEC STACK */
                                                 /* ACCORDING TO THE # OF     */
                                                 /* OPERANDS                  */
                                                 /*                          */
                        ETOP = ETOP - DEGL;
                        IF DEGR > 0 THEN         /* BUILD AN EXEC STACK ENTRY */
                                                 /* FOR THE RESULT            */
                        DO;
                            I, ETOP = ETOP + 1;
                            EXCON(I) = EA ¬= 0;
                            EXTYPE(I) = RESTYPE;
                            EXADD(I) = EA;
                            EXVAL#(I) = V;
                            EXNAM#(I) = 0;
                        END;
                    END;
                END;                             /* END OF SIMPLE OPERATOR    */
TYPE1_OP:
                                                 /* TYPE 1 OPERATOR NOT USED  */
                                                 /*                          */
                CALL CERROR('UNDEFINED OPERATOR TYPE - TYPE 1');
                                                 /*                          */
                                                 /* CASE 2 LOAD OPERATOR      */
                                                 /* MUST BE LOCATION TYPE     */
                                                 /* VARIABLE ON THE TOP OF THE*/
                                                 /* EXEC STACK TO ALLOW A LOAD*/
                                                 /* FROM THE ADDRESS ON TOP   */
                                                 /* OF STACK.                 */
                                                 /*                          */
LOD_OP:
                DO;
                    IF EXTYPE(ETOP) ¬= LOC THEN
                    DO;
                        CALL CERROR('ATTEMPT TO LOAD FROM NON-LOC TYPE');
                        RETURN;
                    END;
                    A = EXADD(ETOP);
                    IF EXCON(ETOP) THEN          /* CHECK FOR VALUE IN THE    */
                                                 /* CONSTANT TABLE            */
                    DO;
                                                 /*                          */
                                                 /* YES...SET VARIABLES TO    */
                                                 /* INDICATE CONSTANT TO BE   */
                                                 /* LOADED                    */
                                                 /*                          */
                        V = CONVAL(A);
                        C = TRUE;
                        TYP = CONTYPE(A);
                    END; ELSE
                    DO;
                                                 /*                          */
                                                 /* NO...FIND IDENTIFIER IN   */
                                                 /* ADDRESS TABLE             */
                                                 /*                          */
                        I = LOOKADD(A);
                        V = ADDVAL(I);
                        IF C THEN                /* CHECK FOR PREVIOUSLY      */
                                                 /* ASSIGNED CLASS NUMBER.    */
                                                 /* YES...MARK LOCATION       */
                        V, ADDVAL(I) = GETVAL#(COMPU);
                        ELSE
                        IF V = 0 THEN            /* NO...CHECK FOR VALUE #    */
                                                 /* ASSIGNED                  */
                        DO;
                                                 /*                          */
                                                 /* NO...ASSIGN BY CODE       */
                                                 /* LOCATION AND FLAG THE CODE*/
                                                 /* FILE                      */
                                                 /*                          */
                            V, ADDVAL(I) = GETVAL#(COMPU);
                            CSFCODE(M_CONTROL) = (CODE & "FFFF0000")|SHL(1, 15);
                                                 /*                          */
                                                 /* MAY WANT TO PUT TYPE IN   */
                                                 /* THIS FIELD LATER.         */
                                                 /*                          */
                        END;
```

```
                                                  /*                            */
                                                  /* MARK TYP AND A FROM ADDRES*/
                                                  /* TABLE AND SET C BY VALUE   */
                                                  /* TABLE PRESENT OR NOT       */
                                                  /*                            */
                    TYP = ADDTYPE(I);
                    A = ADDCON(I);
                    C = A ¬= 0;
                END;
                                                  /*                            */
                                                  /* SET TOP OF EXEC STACK TO   */
                                                  /* LOAD EITHER CONSTANT OR    */
                                                  /* VALUE AS PREVIOUSLY        */
                                                  /* DETERMINED                 */
                                                  /*                            */
                EXNAM#(ETOP) = EXVAL#(ETOP);
                EXCON(ETOP) = C;
                EXTYPE(ETOP) = TYP;
                EXADD(ETOP) = A;
                EXVAL#(ETOP) = V;
            END;
                                                  /*                            */
                                                  /* CASE 3 STORE OPERATOR      */
                                                  /* CAUSES SECOND OPERAND      */
                                                  /* ON STACK TO BE STORED IN   */
                                                  /* LOCATION SPECIFIED BY      */
                                                  /* ADDRESS ON TOP OF STACK.   */
                                                  /*                            */
STORE_OP:
            DO;
                                                  /*                            */
                                                  /* TOP OF EXEC STACK MUST BE  */
                                                  /* LOC                        */
                                                  /*                            */
                IF EXTYPE(ETOP - 1) ¬= LOC THEN
                DO;
                    CALL CERROR('ATTEMPT TO STORE INTO NON-LOC VARIABLE');
                    RETURN;
                END;
                A = EXADD(ETOP - 1);
                                                  /*                            */
                                                  /* IF NO ADDRESS, ASSUME AN   */
                                                  /* ARRAY SUBSCRIPT            */
                                                  /*                            */
                IF A = 0 THEN
                DO;
                    CALL CERROR('ARRAY STORES NOT IMPLEMENTED IN CSF');
                    RETURN;
                END;
                                                  /*                            */
                                                  /* FIND ADDR TABLE LOCATION   */
                                                  /*                            */
                I = LOOKADD(A);
                                                  /*                            */
                                                  /* NOT IN ADDRESS TABLE       */
                                                  /*                            */
                IF I = 0 THEN
                DO;
                    CALL ERROR('ADDRESS TABLE ERROR (2)', 2);
                    RETURN;
                END;
                IF I <= TADDLOC THEN              /* ADDRESS SET IN ANOTHER     */
                                                  /* BBLOCK                     */
                DO;
                                                  /*                            */
                                                  /* BUILD A NEW ADDRESS TABLE  */
                                                  /* ENTRY                      */
                                                  /*                            */
                    N = ADDNAM(I);
                    CALL ENTERADD(A);
                    I = ADDLOC;
                    ADDNAM(I) = N;
                END;
                N = ETOP;
                ALTAD = 1;
                IF EXCON(N) THEN                  /* CONSTANT ASSIGNMENT        */
                DO;                               /* SET ADDRESS TABLE FROM     */
                                                  /* CONSTANT TABLE             */
                    J = EXADD(N);
                    ADDCON(I) = J;
                    ADDVAL(I) = EXVAL#(N);
                    ADDTYPE(I) = CONTYPE(J);
                    ADDNAM(I) = 0;
                END; ELSE
                DO;                               /* NOT A CONSTANT             */
                                                  /* SET ADDRESS TABLE FROM     */
                                                  /* EXEC STACK                 */
                    ADDCON(I) = 0;
```

```
          ADDVAL(I) = EXVAL#(N);
          ADDTYPE(I) = EXTYPE(N);
          ADDNAM(I) = EXNAM#(N);
       END;
       IF RETAIN THEN
       DO;                                  /* IF STACK TOP IS TO BE      */
                                            /* RETAINED, MOVE THE LAST    */
                                            /* ENTRY DOWN ONE             */
          ETOP = ETOP - 1;
          EXCON(ETOP) = EXCON(N);
          EXTYPE(ETOP) = EXTYPE(N);
          EXADD(ETOP) = EXADD(N);
          EXVAL#(ETOP) = EXVAL#(N);
          EXNAM#(ETOP) = EXNAM#(N);
                                            /*                            */
                                            /* NO...DECREMENT STACK TOP   */
                                            /* BY 2                       */
                                            /*                            */
       END; ELSE
       ETOP = ETOP - 2;
    END;
                                            /*                            */
                                            /* CASE 4 STORE AND RETAIN    */
                                            /*                            */
STO_RET_OP:
    DO;
       RETAIN = TRUE;                       /* SIMPLY SET RETAIN FLAG     */
                                            /* AND USE STORE CODING       */
       GO TO STORE_OP;
    END;
                                            /*                            */
                                            /* CASE 5 DELETE              */
                                            /* SIMPLY DELETES THE TOP OF  */
                                            /* EXEC STACK                 */
                                            /*                            */
DEL_OP:
    DO;
       IF ETOP <= 0 THEN
       CALL ERROR('EXECUTION TABLE UNDERFLOW', 1);
       ELSE
       ETOP = ETOP - 1;
    END;
                                            /* CASE 6 DUPLICATE           */
                                            /* DUPLICATES THE TOP ENTRY   */
                                            /* ON THE EXEC STACK          */
                                            /*                            */
DUP_OP:
    DO;
       CALL GET_EXEC;
       I = ETOP - 1;
       EXCON(ETOP) = EXCON(I);
       EXTYPE(ETOP) = EXTYPE(I);
       EXADD(ETOP) = EXADD(I);
       EXVAL#(ETOP) = EXVAL#(I);
       EXNAM#(ETOP) = EXNAM#(I);
    END;
                                            /*                            */
                                            /* CASE 7 CONV                */
                                            /* A TYPE CONVERSION IS       */
                                            /* POSSIBLE. HANDLED IN       */
                                            /* SIMPLIFICATION CODING      */
                                            /*                            */
CONV_OP:
    GO TO SIMPLE_OP;
                                            /*                            */
                                            /* CASE 8 COMMUTATIVE         */
                                            /* A COMMUNITIVE OPERATOR IS  */
                                            /* PRESENT HANDLED IN SIMPLE  */
                                            /* OPERATOR CODING            */
                                            /*                            */
COMM_OP:
    GO TO SIMPLE_OP;
                                            /*                            */
                                            /* CASE 9 EXCHANGE            */
                                            /* EXCHANGES THE TOP 2        */
                                            /* ENTRIES ON THE EXEC STACK  */
                                            /*                            */
EXCHG_OP:
    DO;
       I = ETOP - 1;
       IF I <= 0 THEN
       DO;
          CALL CERROR(' EXCHANGE CAUSES UNDERFLOW');
          RETURN;
       END;
       J = EXCON(ETOP);
       EXCON(ETOP) = EXCON(I);
       EXCON(I) = J;
```

```
                    J = EXTYPE(ETOP);
                    EXTYPE(ETOP) = EXTYPE(I);
                    EXTYPE(I) = J;
                    J = EXADD(ETOP);
                    EXADD(ETOP) = EXADD(I);
                    EXADD(I) = J;
                    J = EXVAL#(ETOP);
                    EXVAL#(ETOP) = EXVAL#(I);
                    EXVAL#(I) = J;
                    J = EXNAM#(ETOP);
                    EXNAM#(ETOP) = EXNAM#(I);
                    EXNAM#(I) = J;
                END;
                                                   /*                            */
                                                   /* CASE 10 COND               */
                                                   /* CONDITIONAL BRANCH         */
                                                   /*                            */
COND_OP:
                DO;
                    IF BRANCHTOG THEN
                    DO;
                                                   /*                            */
                                                   /* IF TOP OF EXEC STACK IS    */
                                                   /* CONSTANT GET RID OF FALSE  */
                                                   /* BRANCH BEFORE EXIT BASIC   */
                                                   /* BLOCK                      */
                                                   /*                            */
                        IF EXCON(ETOP) THEN
                        CALL PROPAGATE(OP, 0);
                    END;
                    ETOP = ETOP - 1;
                    RETURN;
                END;
                                                   /*                            */
                                                   /* CASE 11 UCOND              */
                                                   /* UNCONDITIONAL BRANCH       */
                                                   /*                            */
UCOND_OP:
                RETURN;
                                                   /*                            */
                                                   /* CASE 12 GOSUB              */
                                                   /* BRANCH TO A SUBROUTINE     */
                                                   /* JUMP OUT AFTER SETTING     */
                                                   /* NEXT BLOCK ON CONTROL      */
                                                   /* STACK                      */
                                                   /*                            */
GOSUB_OP:
                DO;
                    CALL INSRT_SUC_BLOCK(M_CONTROL + 1);
                    RETURN;
                END;
                                                   /* CASE 13 RETSUB             */
                                                   /* BRANCH RETURN FROM A SUBR  */
                                                   /* TO ORIGINAL POINT OF GOSUB*/
                                                   /* INVOCATION.                */
                                                   /*                            */
RETSUB_OP:
                RETURN;
            END;                                   /* END OF ALL OPERATOR TYPES  */
        END;                                       /* END OF ELSE (TYP > XIT)    */
                                                   /*                            */
                                                   /* CHECK THAT EXEC STACK HAS  */
                                                   /* NOT BEEN DECREMENTED BELOW*/
                                                   /* ZERO                       */
                                                   /*                            */
        IF ETOP < 0 THEN
        DO;
            CALL ERROR('EXECUTION STACK UNDERFLOW', 1);
            RETURN;
        END;                                       /* DUMP TABLES IF SET         */
        IF TABLETOG THEN
        CALL W_ALL;
    END;                                           /* END OF THE M_CONTROL LOOP */
                                                   /*                            */
                                                   /* THIS RETURN ONLY USED IF   */
                                                   /* THE ENTRY CODE FILE IS     */
                                                   /* EXHAUSTED                  */
                                                   /*                            */
    RETURN;
END BASIC_BLOCK;
            /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
            /*                                              */
            /*            SET_UP_BLOCK                      */
            /* USING THE TOP BLOCK ON THE CONTROL STACK     */
            /* ALL TABLES, ETC., ARE SET TO BE ITS          */
            /* CURRENT OPTIMIZED STATE.                     */
            /* ROUTINES CALLED:                             */
            /*            REMOVEV                           */
            /*            REENTERV                          */
            /*            ENTERADD                          */
            /*                                              */
            /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
```

```
SET_UP_BLOCK: PROCEDURE;
     DECLARE (I,J,K,L,M) FIXED;
     DECLARE (O,V) FIXED;
                                                            /*                            */
                                                            /* FORM POINTER TO STATES     */
                                                            /* TABLE CURRENT BLOCK        */
                                                            /*                            */
     ALTEX = 1;
     ETOP = 0;
                                                            /*                            */
                                                            /* RESTORE THE ADDRESS TABLE  */
                                                            /* FROM THE STATE VECTOR      */
                                                            /*                            */
     ALTVL = 1;
     ALTAD = 1;
     ADDLOC = 0;
                                                            /*                            */
                                                            /* REMOVE ALL VTAB ENTRIES    */
                                                            /* ABOVE HASHBASE             */
                                                            /*                            */
     CALL REMOVEV(HASHBASE);
     I = CSFCODE(CURBLOCK) & "FFFF";
     K = I + 2;
     J = STATES(K);
     K = K + 1;
     I = K + J;
                                                            /*                            */
                                                            /* REBUILD THE CURRENT        */
                                                            /* OPTIMIZE STATE FOR THE     */
                                                            /* BLOCK.                     */
                                                            /*                            */
     DO WHILE I > K;
          I = I - 1;
          J = STATES(I) & "7FFF";
          L = ADDRESS(J);
          IF SHR(L, 15) THEN                                /* EXEC STACK ENTRY           */
                                                            /* ADDRESS = |1| ETOP         */
                                                            /* PUT THE EXEC STACK ENTRY   */
                                                            /* AT THE TOP OF THE EXEC     */
                                                            /* STACK AREA.                */
          DO;
               L = L & "7FFF";
               IF ETOP <= L THEN
               ETOP = L + 1;
               L = MAXEXEC - L;
               EXTYPE(L) = ADDTYPE(J);
               EXVAL#(L) = ADDVAL(J);
               M = ADDCON(J);
               EXCON(L) = SHR(M, 15);
               EXADD(L) = M & "7FFF";
               EXNAM#(L) = ADDNAM(J);
          END; ELSE
          IF L = 0 THEN                                     /* VALUE TABLE ENTRY          */
                                                            /* ADDRESS = 0                */
                                                            /* PUT THE ADDRESS OF THE     */
                                                            /* VTAB ENTRY ON THE EXEC     */
                                                            /* STACK.                     */
          DO;
               O = ADDTYPE(J);
               L = ETOP;
               V = ADDVAL(J);
               J = ADDNAM(J);
               M = OP_DEGL(O);
               DO WHILE M > 0;
                    M = M - 1;
                    ETOP = ETOP + 1;
                    EXVAL#(ETOP) = VTAB(J);
                    J = J - 1;
               END;
                                                            /* BUILD A VALUE TABLE ENTRY  */
                                                            /* FOR EACH TYPE OF           */
                                                            /* EXPRESSION.                */
               M = REENTERV(O, COMPU);
               ETOP = L;
               M = M + 2 + OP_DEGL(O);
               VTAB(M) = V;
          END; ELSE
                                                            /*                            */
                                                            /* ADDRESS TABLE ENTRY ;      */
                                                            /* SIMPLY RESTORES THE ADDRES*/
                                                            /* TABLE FOR THIS BLOCK       */
                                                            /*                            */
          DO;
               CALL ENTERADD(L);
```

```
                ADDTYPE(ADDLOC) = ADDTYPE(J);
                ADDVAL(ADDLOC) = ADDVAL(J);
                ADDCON(ADDLOC) = ADDCON(J);
                ADDNAM(ADDLOC) = ADDNAM(J);
          END;
     END;
                                                                  /*                          *.
                                                                  /* NOW FILL THE EXEC STACK  *.
                                                                  /* MOVES THE EXEC DATA FROM *.
                                                                  /* THE TOP OF EXEC AREA INTO *.
                                                                  /* WORKING EXEC STACK.      *.
                                                                  /*                          *.
     L = ETOP;
     M = MAXEXEC;
     DO WHILE L > 0;
          EXTYPE(L)=EXTYPE(M);
          EXVAL#(L)=EXVAL#(M);
          EXCON(L)=EXCON(M);
          EXADD(L)=EXADD(M);
          EXNAM#(L)=EXNAM#(M);
          M = M - 1;
          L = L - 1;
     END;
END SET_UP_BLOCK;
                 /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                 /*                                            */
                 /*                 SWAP_BLOCKS                */
                 /* THIS ROUTINE MOVES THE ENTRIES IN THE      */
                 /* CONTROL LIST.  THE ENTRIES MOVED ARE AT    */
                 /* ADDRESS A AND B IN THE CONTROL STACK.      */
                 /*                                            */
                 /*<><><><><><><><><><><><><><><><><><><><><><><>*/
SWAP_BLOCKS: PROCEDURE (A, B);
     DECLARE (A, B, TEMFIX) FIXED, TEM8 BIT(8);
          TEMFIX = CONTROL(B);
          CONTROL(B) = CONTROL(A);
          CONTROL(A) = TEMFIX;
          TEM8 = SAV_TOG(B);
          SAV_TOG(B) = SAV_TOG(A);
          SAV_TOG(A) = TEM8;
END SWAP_BLOCKS;
                 /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                 /*                                            */
                 /*              ORDER_BLOCK_LIST              */
                 /* THIS ROUTINE SETS UP THE CONTROL STACK     */
                 /* BY DIFFERENT ORDERING ALGORITHMS           */
                 /* DEPENDENT UPON SELECT_FLAG.                */
                 /*          0 = CANNONICAL (LIFO)             */
                 /*          1 = STEEPEST DESCENT              */
                 /*              (HEURISTICALLY THIS IS DONE   */
                 /*              BY TAKING THE BLOCK WITH THE  */
                 /*              LEAST # OF STATES AND MOVING  */
                 /*              IT TO BE PROCESSED NEXT).     */
                 /*          2 = FIFO                          */
                 /*          3 = SELECTS THE BLOCK WITH THE    */
                 /*              MAXIMUM CURRENT STATE.        */
                 /* ROUTINES CALLED:                           */
                 /*              SWAP_BLOCKS                   */
                 /*                                            */
                 /*<><><><><><><><><><><><><><><><><><><><><><><>*/
ORDER_BLOCK_LIST: PROCEDURE;
     DECLARE (CODEADR, BLKHDADRA, BLKHDADRB, STATEA, STATEB, NUMSTATESA,
              NUMSTATESB, I, L, J, K, FOUND,DATA) FIXED;
     DO CASE SELECT_FLAG;
CANNONICAL:
                                                                  /*                          */
                                                                  /* CANNONICAL REQUESTED     */
                                                                  /* CONTROL STACK IS ALREADY */
                                                                  /* BUILT PROPERLY FOR LIFO  */
                                                                  /*                          */
     RETURN;
                                                                  /*                          */
                                                                  /* STEEPEST DESCENT REQUESTED*/
                                                                  /*                          */
STEEPEST_DESCENT:
     DO;                                                          /* ONLY ONE ENTRY ON STACK  */
          IF CTOP <= 1 THEN                                       /* YES...EXIT STACK MUST BE */
          RETURN;                                                 /* RIGHT                    */
                                                                  /*                          */
                                                                  /* NO...SORT STACK INTO     */
                                                                  /* DESCENDING ORDER         */
                                                                  /*                          */
          J = 1;
          DO WHILE J < CTOP;
               CODEADR = CONTROL(J);
                                                                  /*                          */
                                                                  /* GET THE STATE POINTER    */
                                                                  /*                          */
```

```
                CODEADR = CODEADR & "7FFF";
                BLKHDADRA = CSFCODE(CODEADR);
                BLKHDADRA = BLKHDADRA & "7FFF";
                CODEADR = CONTROL(J + 1);
                CODEADR = CODEADR & "7FFF";
                BLKHDADRB = CSFCODE(CODEADR);
                BLKHDADRB = BLKHDADRB & "7FFF";
                STATEA = CSFCODE(BLKHDADRA);
                STATEA = STATEA & "7FFF";
                STATEB = CSFCODE(BLKHDADRB);
                STATEB = STATEB & "7FFF";
                STATEA = STATEA + 2;
                STATEB = STATEB + 2;
                NUMSTATESA = STATES(STATEA);
                NUMSTATESB = STATES(STATEB);
                                                                /*                            */
                                                                /* IN DESCENDING ORDER        */
                                                                /*                            */
                IF NUMSTATESA < NUMSTATESB THEN
                DO;
                    CALL SWAP_BLOCKS(J, J + 1);
                    K = 1;
                    L = J;
                                                                /*                            */
                                                                /* BUBBLE B TO PROPER PLACE   */
                                                                /*                            */
                    I = J - 1;
                    DO WHILE I > 0;
                        CODEADR = CONTROL(I);
                        CODEADR = CODEADR & "7FFF";
                        BLKHDADRA = CSFCODE(CODEADR);
                        BLKHDADRA = BLKHDADRA & "7FFF";
                        STATEA = CSFCODE(BLKHDADRA);
                        STATEA = STATEA & "7FFF";
                        STATEA = STATEA + 2;
                        NUMSTATESA = STATES(STATEA);
                        IF NUMSTATESA < NUMSTATESB THEN
                        DO;
                            CALL SWAP_BLOCKS(I, L);
                            L = I;
                        END;
                        I = I - 1;
                    END;                                        /* END WHILE K                */
                END;                                            /* END FIRST A < B            */
                J = J + 1;
            END;                                                /* DO WHILE                   */
        END;                                                    /* STEEPEST DESCENT           */
    FIRST_IN_FIRST_OUT:
        DO;
                                                                /*                            */
                                                                /* IF ONLY ONE ENTRY, LIST IS*/
                                                                /* ALREADY GOOD               */
                                                                /*                            */
            IF CTOP <= 1 THEN
            RETURN;
            FOUND = FALSE;
            J = 1;
            DO WHILE ¬ FOUND;
                DATA = CONTROL(J);
                                                                /*                            */
                                                                /* IS BLOCK GOOD              */
                                                                /*                            */
                IF ¬(SHR(DATA, 31) & "1")THEN
                DO;                                             /* YES...BUBBLE BLOCK TO TOP */
                    FOUND = TRUE;
                    K = TRUE;
                    DO WHILE K;
                        CALL SWAP_BLOCKS(J, J + 1);
                        IF J = (CTOP - 1) THEN
                        K = FALSE;
                        J = J + 1;
                    END;
                END;
                J = J + 1;
                IF J >= CTOP THEN
                RETURN;
            END;
        END;
    MAX_POOL:
        DO;                                                     /* ONLY ONE ENTRY ON STACK    */
            IF CTOP <= 1 THEN                                   /* YES...EXIT STACK MUST BE   */
            RETURN;                                             /* RIGHT                      */
                                                                /*                            */
                                                                /* NO...SORT STACK INTO       */
                                                                /* ASSENDING ORDER            */
                                                                /*                            */
            J = 1;
            DO WHILE J < CTOP;
```

```
                                                  /*                              */
                                                  /* GET THE STATE POINTER        */
                                                  /*                              */
            CODEADR = CONTROL(J);
            CODEADR = CODEADR & "7FFF";
            BLKHDADRA = CSFCODE(CODEADR);
            BLKHDADRA = BLKHDADRA & "7FFF";
            CODEADR = CONTROL(J + 1);
            CODEADR = CODEADR & "7FFF";
            BLKHDADRB = CSFCODE(CODEADR);
            BLKHDADRB = BLKHDADRB & "7FFF";
            STATEA = CSFCODE(BLKHDADRA);
            STATEA = STATEA & "7FFF";
            STATEB = CSFCODE(BLKHDADRB);
            STATEB = STATEB & "7FFF";
            STATEA = STATEA + 2;
            STATEB = STATEB + 2;
            NUMSTATESA = STATES(STATEA);
            NUMSTATESB = STATES(STATEB);
            IF NUMSTATESA > NUMSTATESB THEN
            DO;
                CALL SWAP_BLOCKS(J, J + 1);
                K = 1;
                L = J;
                                                  /*                              */
                                                  /* BUBBLE B TO PROPER PLACE     */
                                                  /*                              */
                I = J - 1;
                DO WHILE I > 0;
                    CODEADR = CONTROL(I);
                    CODEADR = CODEADR & "7FFF";
                    BLKHDADRA = CSFCODE(CODEADR);
                    BLKHDADRA = BLKHDADRA & "7FFF";
                    STATEA = CSFCODE(BLKHDADRA);
                    STATEA = STATEA & "7FFF";
                    STATEA = STATEA + 2;
                    NUMSTATESA = STATES(STATEA);
                    IF NUMSTATESA > NUMSTATESB THEN
                    DO;
                        CALL SWAP_BLOCKS(I, L);
                        L = I;
                    END;
                    I = I - 1;
                END;                              /* END WHILE K                  */
            END;                                  /* END FIRST A < B              */
            J = J + 1;
        END;                                      /* DO WHILE                     */
    END;                                          /* MAX POOL                     */
    END;                                          /* OF ALL CASES                 */
    RETURN;
END ORDER_BLOCK_LIST;
              /*<><><><><><><><><><><><><><><><><><><><><><><>*/
              /*                                              */
              /*              BUILD_INPUT_POOL                */
              /* 1..MOVES THE ADDR TABLE ENTRIES TO THE       */
              /*    TOP OF THE TABLE, THE SAVE AREA FOR       */
              /*    THE CURRENT OPTIMIZED STATE FOR EACH      */
              /*    BLOCK.  AN ENTRY IS MADE FOR EACH         */
              /*    CLASS.                                    */
              /*    NOTE:  IF AN ADDRESS ENTRY CORRESPONDS*/
              /*    TO A VTAB (EXPRESSION), THE VTAB          */
              /*    ENTRY IS COPIED TO TOP OF VTAB AND        */
              /*    ADDNAM IS SET TO POINT TO IT.             */
              /* 2..BUILDS THE STATE TABLE FOR THIS BLOCK */
              /*    AS FOLLOWS:                               */
              /*          X     |    0                        */
              /*          X+1   |    0                        */
              /*          X+2   |    NUMBER OF VALUE NUMBERS  */
              /*          X+3   |    ADDR IN TOP OF ADDR TAB */
              /*                |    OF THE VALUE NUMBER      */
              /*                |    ENTRY, IE. EQUIVALENCE   */
              /*                |    CLASS ENTRY              */
              /*          ETC   |                             */
              /*          X+I   |                             */
              /*    RETURNS ADDRESS OF X - 2                  */
              /* ROUTINES CALLED:                             */
              /*                GET_ST                        */
              /*                LOOKADD                       */
              /*                GET_ADDR                      */
              /*                GET_VTOPR                     */
              /*                                              */
              /*<><><><><><><><><><><><><><><><><><><><><><><>*/
BUILD_INPUT_POOL: PROCEDURE FIXED;
    DECLARE (TOPADR, STPNT, VARCONIND, ACTADR, CONS, VALUE, VALPNT,
            CNT, OPTYP, OPNT, NAMPNT) FIXED;
                                                  /*                              */
                                                  /* ADV STATE POINTER TWO        */
                                                  /* LOCATIONS                    */
```

```
                                                             /*                           */
      CALL GET_ST;
      CALL GET_ST;
      CNT = 0;
      TOPADR = ADDLOC;
      CALL GET_ST;
      STPNT = STLOC;
                                                             /*                           */
                                                             /* MOVE ALL ADDRESS TABLE    */
                                                             /* ENTRIES.                  */
                                                             /*                           */
      DO WHILE TOPADR > 0;
           VARCONIND = ADDRESS(TOPADR);
           IF VARCONIND = 0 THEN
           ACTADR = TOPADR;
           ELSE ACTADR = LOOKADD(VARCONIND);
           IF ACTADR = TOPADR THEN                           /* TOP-MOST ENTRY IN ADDRESS */
                                                             /* TABLE.                    */
           DO;
                CONS = ADDCON(ACTADR);
                VALUE = ADDVAL(ACTADR);
                VALPNT = ADDNAM(ACTADR);
                IF VALUE ¬= 0 THEN                           /* A VALUE # IS ASSIGNED     */
                                                             /* IMPLIES USEFUL INFORMATION*/
                DO;                                          /* MAKE AN ENTRY IN THE STATE*/
                                                             /* VECTOR SAVE AREA          */
                     CALL GET_ST;
                     CNT = CNT + 1;
                     CALL GET_ADDR;
                     ADDRESS(ADDLOCR) = VARCONIND;
                     ADDCON(ADDLOCR) = CONS;
                     ADDVAL(ADDLOCR) = VALUE;
                                                             /*                           */
                                                             /* MUST HAVE A VALUE TABLE   */
                                                             /* ENTRY FOR THE ADDRESS     */
                                                             /* ENTRY IF NO ADDRESS #     */
                                                             /* ADDNAM IS POINTER TO VTAB */
                                                             /*                           */
                     IF VARCONIND = 0 THEN
                     DO;
                          OPTYP = ADDTYPE(ACTADR);
                          OPNT = OP_DEGL(OPTYP);
                          VALPNT = VTOPR - 1;
                          NAMPNT = ADDNAM(ACTADR) + 1;
                                                             /*                           */
                                                             /* MOVE THE VALUE TABLES     */
                                                             /* ENTRY FROM VTAB ENTRY TO  */
                                                             /* TOP OF VTAB (SAVE AREA)   */
                                                             /*                           */
                          DO WHILE OPNT > 0;
                               OPNT = OPNT - 1;
                               NAMPNT = NAMPNT + 1;
                               CALL GET_VTOPR;
                               VTAB(VTOPR) = VTAB(NAMPNT);
                          END;
                     END;
                                                             /*                           */
                                                             /* SET NEW POINTER IN ADDNAM */
                                                             /*                           */
                     ADDNAM(ADDLOCR)= VALPNT;
                     ADDTYPE(ADDLOCR) = ADDTYPE(ACTADR);
                     STATES(STLOC) = ADDLOCR;
                END;
           END;
           TOPADR = TOPADR - 1;
      END;
                                                             /*                           */
                                                             /* MARK THE # OF CLASSES THIS*/
                                                             /* STATE.                    */
                                                             /*                           */
      STATES(STPNT) = CNT;
      RETURN STPNT - 2;
END BUILD_INPUT_POOL;
```

```
/*<><><><><><><><><><><><><><><><><><><><><><><>*/
/*                                               */
/*              PERFORM_MEET_OP                  */
/* THIS ROUTINE PERFORMS THE MEET OPERATION      */
/* UPON THE CURRENT POOL VS THE INPUT POOL.      */
/* FOR THE OPTIMIZATION FUNCTIONS USED, THE      */
/* MEET OPERATION IS DEFINED TO BE:              */
/*       P   = P   (AND) P                       */
/*        I     C         O                      */
/* WHERE      PC = CURRENT STATE OF AN           */
/*                 IMMEDIATE SUCCESSOR           */
/*                 BLOCK                         */
/*            PI = INPUT POOL TO BE USED         */
/*                 TO PROCESS THE SUCCESSOR      */
/*                 BLOCK                         */
/*            PO = OUTPUT POOL                   */
/* RETURNS TRUE IF CURRENT POOL DECREASES        */
/* IN SIZE, FALSE IF NO CHANGE IN CURRENT        */
/* POOL.                                         */
/* ROUTINES WITHIN PROCEDURE:                    */
/*              W_OCCURANCE                      */
/*              W_REFINEMENT                     */
/*              INITIAL_PMO                      */
/*              LOOK_VARIABLE                    */
/*              BUILD_EXP_ENTRY                  */
/*              SRCH_OCCURANCE_LIST              */
/*              BUILD_CLASS_OCCURANCE_LIST       */
/*              BUILD_NEW_SIM_VAR_ENTRY          */
/*              ASSIGN_REFINEMENT                */
/*              SEARCH_REFINEMENT_LIST           */
/*              DELETE_OCCURANCE                 */
/*              PERMUTE_REFINEMENT               */
/*              SET_SEARCH_LIST                  */
/* ROUTINES CALLED:                              */
/*              INITIAL_PMO                      */
/*              LINE_FEED                        */
/*              PRINT_OUTPUT_POOL                */
/*              W_STATE                          */
/*              ERROR                            */
/*              BUILD_CLASS_OCCURANCE_LIST       */
/*              W_OCCURANCE                      */
/*              DELETE_OCCURANCE                 */
/*              LOOK_VARIABLES                   */
/*              SEARCH_REFINEMENT_LIST           */
/*              ASSIGN_REFINEMENT                */
/*              BUILD_NEW_SIM_VAR_ENTRY          */
/*              W_REFINEMENT                     */
/*              SRCH_OCCURANCE_LIST              */
/*              BUILD_EXP_ENTRY                  */
/*                                               */
/*<><><><><><><><><><><><><><><><><><><><><><><>*/
```

```
PERFORM_MEET_OP: PROCEDURE(BLKADR) BIT(1);
   DECLARE     CUR_STATE_CHGD BIT(1),                           /* TRUE IF STATE HAS BEEN     */
                                                                /* REDUCED.                   */
               TOT_CLASS_CNT FIXED,                             /* HOLDS TOTAL NUMBER OF      */
                                                                /* ENTRIES IN CURRENT         */
                                                                /* OPTIMIZED POOL             */
               SIMP_VARIABLES_PRES FIXED,                       /* IF SIMP VARIABLES PRESENT */
               MAX_ELEMENTS       LITERALLY    '90',            /* MAX SIZE OF OCCURANCE LIST*/
               OCCTOP        FIXED,                             /* HOLDS NUMBER OF CLASSES IN*/
                                                                /* THE OCCURANCE LIST         */
               OCCNUM        (MAX_ELEMENTS) BIT(16),            /* HOLDS # OF ELEMENTS IN     */
                                                                /* EACH CLASS                 */
               OCCVAL        (MAX_ELEMENTS) BIT(16),            /* HOLDS CLASS VALUE#         */
               MAX_REFINEMENTS  LITERALLY    '90',              /* MAX SIZE OF REFINEMENT     */
               RTOP          FIXED,                             /* TOP OF THE REFINEMENT LIST*/
               REFNVAL       (MAX_REFINEMENTS) BIT(16),/*HOLDS REFINED VALUE NUMBER*/
               REFCVAL       (MAX_REFINEMENTS) BIT(16),/*HOLDS CURRENT POOL VALUE# */
               REFIVAL       (MAX_REFINEMENTS) BIT(16),/*HOLDS VALUE NUMBER IN     */
                                                                /* INPUT POOL                 */
                                                                /* CLASS MAPPINGS  V(V1, V2) */
                                                                /* WHERE  V1 = VAL# IN        */
                                                                /*             CURRENT POOL   */
                                                                /*        V2 = VAL# IN INPUT */
                                                                /*             POOL           */
                                                                /*         V = NEW CLASS #    */
               REFOPERS(MAX_REFINEMENTS) BIT(16),               /* NEW CLASS NUMBER FOR AN    */
                                                                /* EXPRESSION OPERAND         */
               INPOPERS(MAX_REFINEMENTS) BIT(16),               /* OPERAND TO SEARCH FOR IN   */
                                                                /* THE INPUT EXPRESSION       */
               OPERS(MAX_REFINEMENTS) BIT(16),                  /* CURRENT EXP OPERANDS       */
               STOP          FIXED,                             /* COUNTER FOR EXPRESSION     */
                                                                /* CLASS INTERSECTION TABLE   */
               SPOINT        FIXED,                             /* POINTER FOR EXPRESSION     */
                                                                /* CLASS INTERSECTION TABLE   */
               SVPNT         FIXED,
               NEW_ST_COUNTER  FIXED,                           /* COUNTER FOR THE NUMBER OF */
                                                                /* CLASSES IN THE MERGED      */
                                                                /* STATE                      */
               NEW_ST_PT     FIXED,                             /* POINTER FOR THE NEW STATE */
               STPT          FIXED,                             /* CURRENT STATE POINTER      */
               STCOUNTER     FIXED,                             /* LOOP CONTROL FOR CURRENT   */
               OPTIMADR      FIXED,                             /* ADR OF CURRENT CLASS       */
               INPUTADR      FIXED,                             /* ADR OF THE INPUT POOL EXP */
                                                                /* BEING PROCESSED            */
               OPTIMVAR      FIXED,                             /* INDICATOR AS TO TYPE OF    */
                                                                /* CURRENT POOL ENTRY         */
                                                                /*   0 = EXPRESSION           */
                                                                /*   ¬= 0 = SIMPLE VARIABLE   */
               OPTIMVAL      FIXED
```

```
            OPTIMVAL      FIXED,                          /* CURRENT POOL VALUE#          */
            REFINADR      FIXED,
            INPUTVAL      FIXED,                          /* INPUT POOL VALUE #           */
            EXPAOR        FIXEO,                          /* ADR OF THE OPTIM INFO IN     */
                                                          /* TOP OF VTAB                  */
            OCCADR        FIXED,                          /* ADR OF VALUE # IN THE        */
                                                          /* OCCURENCE LIST               */
            NUMOP         FIXED,                          /* NUMBER OF OPERANOS           */
            BLKADR        FIXEO,                          /* ADR OF BLOCK HEADER FOR      */
                                                          /* THE CURRENT POOL             */
            OPVAL         FIXEO,                          /* CURRENT POOL OPERANO VAL#    */
            INRES         FIXED,                          /* INPUT POOL RESULT VAL#       */
            CURES         FIXEO,                          /* CURRENT POOL RESULT VAL#     */
            REFADR        FIXEO,                          /* REFINEMENT TABLE ADR         */
            SAVST         FIXED,                          /* STORAGE FOR STATE POINTER    */
            SAVCNT        FIXED,                          /* STORAGE FOR STATE COUNT      */
            OPER          FIXEO,                          /* CURRENT POOL OPERATOR        */
            TSTOPER       FIXEO,                          /* INPUT POOL OPERATOR          */
            COMMOP        BIT(1),                         /* TRUE IF CURRNT POOL EXP IS   */
                                                          /* COMMUNITIVE                  */
            CURESTACK     BIT(1),                         /* TRUE IF CURRENT POOL         */
                                                          /* SIMPLE VARIABLE IS ON THE    */
                                                          /* EXEC STACK                   */
            ESTACK        BIT(1),                         /* TRUE IF INPUT POOL SIMPLE    */
                                                          /* VARIABLE IS ON EXEC STACK    */
            FIRSTCALLR    BIT(1),
                                                          /* USED TO PERMUTE REFINEMENT   */
                                                          /* TABLE..INDICATES FIRSTPASS   */
            BOPAR         FIXEO,                          /* POINTER FOR PERMUTE REF.     */
            BOPBR         FIXEO,                          /* POINTER FOR PERMUTE REF.     */
            INPTYP        FIXED,                          /* TYPE OF INPUT POOL ENTRY     */
            OPTTYP        FIXEO,                          /* TYPE OF CURRENT POOL ENTRY   */
                                                          /* MISC POINTERS, COUNTS, ETC   */
            FOUNO         BIT(1),
            COUNT         FIXEO,
            TEMP          FIXEO,
            I             FIXEO,
            FINO          BIT(1);
             /*<><><><><><><><><><><><><><><><><><><><><><><>*/
             /*                                              */
             /*                 W_OCCURRENCE                 */
             /* THIS ROUTINE OUTPUTS THE CLASS OCCURANCE     */
             /* LIST.                                        */
             /* ROUTINES CALLEO:                             */
             /*              LINE_FEED                       */
             /*              PAD                             */
             /*                                              */
             /*<><><><><><><><><><><><><><><><><><><><><><><>*/
W_OCCURRENCE: PROCEDURE;
     DECLARE I FIXED, T CHARACTER;
     OUTPUT = ' CLASS OCCURENCE LIST:';
     IF PUNCHTOG THEN
     OUTPUT(2) = ' CLASS OCCURRENCE LIST:';
     IF OCCTOP <= 1 THEN
     DO;
          OUTPUT = ' (EMPTY)';
          IF PUNCHTOG THEN
          OUTPUT(2) = ' (EMPTY)';
          CALL LINE_FEED(2);
          RETURN;
     ENO;
     CALL LINE_FEEO(1);
     OUTPUT = ' ADDR        CLASS NUMBER      NUMBER OF OCCURRENCES';
     IF PUNCHTOG THEN
     OUTPUT(2) = ' ADDR        CLASS NUMBER      NUMBER OF OCCURRENCES';
     I = 1;
     OO WHILE I < OCCTOP;
          T = ' '|| I;
          T = PAO(T, 15, RIGHT);
          T = T || OCCVAL(I);
          T = PAD(T, 32, RIGHT);
          T = T || OCCNUM(I);
          OUTPUT = T;
          IF PUNCHTOG THEN
          OUTPUT(2) = T;
          I = I + 1;
     ENO;
     CALL LINE_FEEO(2);
ENO W_OCCURRENCE;
             /*<><><><><><><><><><><><><><><><><><><><><><><>*/
             /*                                              */
             /*                 W_REFINEMENT                 */
             /* THIS ROUTINE OUTPUTS THE CLASS REFINEMENT    */
             /* TABLE.                                       */
             /* ROUTINES CALLED:                             */
             /*              LINE_FEED                       */
             /*              PAD                             */
             /*                                              */
             /*<><><><><><><><><><><><><><><><><><><><><><><>*/
W_REFINEMENT: PROCEOURE;
     DECLARE I FIXED, T CHARACTER;
     OUTPUT = ' CLASS REFINEMENT TABLE:';
```

```
      IF PUNCHTOG THEN
      OUTPUT(2) = ' CLASS REFINEMENT TABLE:';
      IF RTOP <= 1 THEN
      DO;
            OUTPUT = ' (EMPTY)';
            IF PUNCHTOG THEN
            OUTPUT(2) = ' (EMPTY)';
            CALL LINE_FEED(2);
            RETURN;
      END;
      CALL LINE_FEED(1);
      OUTPUT = ' LOC          NEW VALUE#      CURRENT VALUE#       INPUT VALUE#';
      IF PUNCHTOG THEN
      OUTPUT(2) = ' LOC          NEW VALUE#      CURRENT VALUE#       INPUT VALUE#';
      CALL LINE_FEED(1);
      I = 1;
      DO WHILE I < RTOP;
            T = ' '|| I;
            T = PAD(T, 15, RIGHT);
            T = T || REFNVAL(I);
            T = PAD(T, 30, RIGHT);
            T = T || REFCVAL(I);
            T = PAD(T, 47, RIGHT);
            T = T || REFIVAL(I);
            OUTPUT = T;
            IF PUNCHTOG THEN
            OUTPUT(2) = T;
            I = I + 1;
      END;
      CALL LINE_FEED(2);
END W_REFINEMENT;
                 /*<><><><><><><><><><><><><><><><><><><><><><>*/
                 /*                                            */
                 /*                 INITIAL_PMO                */
                 /* INITIALIZATION ROUTINE FOR THE PERFORM     */
                 /* MEET OPERATION ROUTINE.                    */
                 /*                                            */
                 /*<><><><><><><><><><><><><><><><><><><><><><>*/
INITIAL_PMO: PROCEDURE;
      DECLARE I FIXED;
      CUR_STATE_CHGD = FALSE;
      SAVST, STPT, STCOUNTER = (CSFCODE(BLKADR) & "FFFF") + 2;
      SAVCNT, STCOUNTER = STATES(STPT);
      OCCTOP = 1;
      DO I = 0 TO (MAX_ELEMENTS -1);
            OCCVAL(I) = 0;
            OCCNUM(I) = 0;
      END;
      RTOP = 1;
      DO I = 0 TO (MAX_REFINEMENTS - 1);
            REFNVAL(I) = 0;
            REFCVAL(I) = 0;
            REFIVAL(I) = 0;
      END;
      TOT_CLASS_CNT = 0;
      SIMP_VARIABLES_PRES = 0;
END INITIAL_PMO;
                 /*<><><><><><><><><><><><><><><><><><><><><><>*/
                 /*                                            */
                 /*                LOOK_VARIABLES              */
                 /* THIS ROUTINE LOOKS FOR AN IDENTIFIER IN    */
                 /* THE ADDRESS TABLES.  AN INDICATOR IS SET   */
                 /* DEPENDING ON WHETHER THE FOUND IDENTIFIER*/
                 /* IS OR IS NOT FROM THE EXECUTION STACK      */
                 /*                                            */
                 /*<><><><><><><><><><><><><><><><><><><><><><>*/
LOOK_VARIABLE: PROCEDURE(SRCHVAR) FIXED;
      DECLARE(SRCHVAR, FNDADR, CURADR, TSTVAR) FIXED, EBIT BIT(1);
      CURADR = ADDLOC;
      FNDADR = 0;
      ESTACK = FALSE;
      DO WHILE (CURADR > 0) & (FNDADR = 0);
            TSTVAR = ADDRESS(CURADR);
            IF TSTVAR ¬= 0 THEN
            DO;
                  EBIT = SHR(TSTVAR, 15) & "1";
                  IF ¬ EBIT THEN                      /* IS ENTRY FROM EXEC STACK   */
                  DO;
                                                      /*                            */
                                                      /* NO...                      */
                                                      /*                            */
                        IF TSTVAR = SRCHVAR THEN
                        DO;
                              FNDADR = CURADR;
                              ESTACK = FALSE;
                        END;
                  END; ELSE
                  DO;
```

```
                                                         /*                              */
                                                         /* YES...                       */
                                                         /*                              */
                    TSTVAR = ADDCON(CURADR) & "7FFF";
                    IF TSTVAR = SRCHVAR THEN
                    DO;
                         FNDADR = CURADR;
                         ESTACK = TRUE;
                    END;
               END;
          END;
          CURADR = CURADR - 1;
     END;
     RETURN FNDADR;
END LOOK_VARIABLE;
               /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
               /*                                                */
               /*             BUILD_EXP_ENTRY                    */
               /* BUILDS AN EXPRESSION ENTRY FOR THE NEW         */
               /* CURRENT STATE.                                 */
               /*                                                */
               /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
BUILD_EXP_ENTRY: PROCEDURE (STADR, RESADR, INADR);
     DECLARE(STADR, RESADR, TEMP, VPT, RESVAL, INADR, CONIN, CONCUR) FIXED;
                                                         /*                              */
                                                         /* ADVANCE THE STATE COUNT      */
                                                         /*                              */
     STATES(NEW_ST_COUNTER) = STATES(NEW_ST_COUNTER) + 1;
     NEW_ST_PT = NEW_ST_PT + 1;
     STATES(NEW_ST_PT) = STADR;
     RESVAL = REFNVAL(RESADR);
     CONCUR = ADDCON(STADR);
     CONIN = ADDCON(INADR);
                                                         /*                              */
                                                         /* IF CONSTANT INDICATOR        */
                                                         /* CHANGE STATE HAS BEEN        */
                                                         /* ALTERED:  I.E. A, 2 => A     */
                                                         /*                              */
     IF CONCUR ¬= 0 THEN
     DO;
          IF CONCUR ¬= CONIN THEN
          DO;
               ADDCON(STADR) = 0;
               CUR_STATE_CHGD = TRUE;
          END;
     END;
                                                         /*                              */
                                                         /* SET IN NEW RESULT VALUE #    */
                                                         /*                              */
     ADDVAL(STADR) = RESVAL;
     VPT = ADDNAM(STADR);
     TEMP = NUMOP;
     DO WHILE NUMOP > 0;
                                                         /*                              */
                                                         /* SET IN NEW OPERAND VALUE #   */
                                                         /*                              */
          VTAB(VPT) = REFOPERS(SPOINT);
          SPOINT = SPOINT + 1;
          VPT = VPT - 1;
          NUMOP = NUMOP - 1;
     END;
     NUMOP = TEMP;
END BUILD_EXP_ENTRY;
               /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
               /*                                                */
               /*           SRCH_OCCURANCE_LIST                  */
               /* ROUTINE SEARCHES FOR A VALUE # IN THE          */
               /* CLASS OCCURANCE LIST.  RETURNS 0 IF NOT        */
               /* FOUND OR ADDRESS IF FOUND.                     */
               /*                                                */
               /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
SRCH_OCCURANCE_LIST: PROCEDURE(VAL#) FIXED;
     DECLARE (VAL#, SRCH_LOC, FOUND) FIXED;
     FOUND = 0;
     SRCH_LOC = 1;
     DO WHILE (SRCH_LOC < OCCTOP) & (FOUND = 0);
          IF VAL# = OCCVAL(SRCH_LOC) THEN
          FOUND = SRCH_LOC;
          SRCH_LOC = SRCH_LOC + 1;
     END;
     RETURN FOUND;
END SRCH_OCCURANCE_LIST;
               /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
               /*                                                */
               /*          BUILD_CLASS_OCCURANCE_LIST            */
               /* BUILDS THE CLASS OCCURANCE LIST BASED          */
               /* UPON THE CURRENT OPTIMIZED POOL FOR THE        */
               /* BLOCK AT BLKADR.                               */
               /*                                                */
               /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
```

```
BUILD_CLASS_OCCURANCE_LIST: PROCEDURE;
     DECLARE (OPSTPT, NUM_ENTRY, CURADR, ENTYP, EXPADR,
             SRCH_VAL#, OCCADR, ADDRS, NUMOP, EXPTYP, I) FIXED;
     OPSTPT =(CSFCODE(BLKADR) & "FFFF") + 2;        /*POINT AT STATE COUNT  */
                                                    /* AND SET UP TO BUILD      */
                                                    /* OCCURANCE LIST           */
     NUM_ENTRY = STATES(OPSTPT);
     DO WHILE NUM_ENTRY > 0;                        /* PROCESS ALL CURRENT STATES*/
          NUM_ENTRY = NUM_ENTRY - 1;                /* DECREMENT ENTRY COUNT AND */
                                                    /* POINT AT STATE TO BE      */
                                                    /* PROCESSED                 */
          OPSTPT = OPSTPT + 1;
          CURADR = STATES(OPSTPT);
          ENTYP, ADDRS = ADDRESS(CURADR);
          IF ENTYP = 0 THEN                         /* IS THIS AN EXPRESSION     */
                                                    /* ENTRY.                    */
          DO;
                                                    /*                           */
                                                    /* YES...SET POINTERS TO VTAB*/
                                                    /*                           */
               EXPADR = ADDNAM(CURADR);
               SRCH_VAL# = ADDVAL(CURADR);
               OCCADR = SRCH_OCCURANCE_LIST(SRCH_VAL#);
               IF OCCADR = 0 THEN
                                                    /*                           */
                                                    /* WAS RESULT VALUE # IN THE */
                                                    /* OCCURANCE LIST            */
                                                    /*                           */
               DO;
                    OCCADR = OCCTOP;
                                                    /*                           */
                                                    /* NO...SET IT IN            */
                                                    /*                           */
                    OCCTOP = OCCTOP + 1;
                    OCCVAL(OCCADR) = SRCH_VAL#;
                    OCCNUM(OCCADR) = 1;
               END; ELSE
               OCCNUM(OCCADR) = OCCNUM(OCCADR) + 1;
               TOT_CLASS_CNT = TOT_CLASS_CNT + 1;
          END; ELSE                                 /* ENTYP ¬= 0                */
          DO;                                       /* WE HAVE A SIMPLE VARIABLE */
                                                    /* FOR EXAMPLE, A CONSTANT.  */
               SIMP_VARIABLES_PRES = SIMP_VARIABLES_PRES + 1;
               SRCH_VAL# = ADDVAL(CURADR);
               OCCADR = SRCH_OCCURANCE_LIST(SRCH_VAL#);
                                                    /*                           */
                                                    /* IS THE SIMPLE VARIABLE    */
                                                    /* VALUE # IN OCCURANCE LIST */
                                                    /*                           */
               IF OCCADR = 0 THEN
               DO;
                                                    /*                           */
                                                    /* NO...SET IT IN            */
                                                    /*                           */
                    OCCADR = OCCTOP;
                    OCCTOP = OCCTOP + 1;
                    OCCVAL(OCCADR) = SRCH_VAL#;
                    OCCNUM(OCCADR) = 1;
               END; ELSE
               OCCNUM(OCCADR) = OCCNUM(OCCADR) + 1;
               TOT_CLASS_CNT = TOT_CLASS_CNT + 1;
          END;                                      /* SIMPLE VARIABLE           */
     END;                                           /* NUM_ENTRY > 0             */
END BUILD_CLASS_OCCURANCE_LIST;
                 /*<><><><><><><><><><><><><><><><><><><><><><>*/
                 /*                                            */
                 /*          BUILD_NEW_SIM_VAR_ENTRY           */
                 /* THIS ROUTINE BUILDS A NEW SIMPLE VARIABLE*/
                 /* CURRENT STATE ENTRY.                       */
                 /*                                            */
                 /*<><><><><><><><><><><><><><><><><><><><><><>*/
BUILD_NEW_SIM_VAR_ENTRY: PROCEDURE (REFADR, STADR, INADR);
     DECLARE (REFADR, INADR, STADR, CONIN,CONCUR) FIXED;
     STATES(NEW_ST_COUNTER) = STATES(NEW_ST_COUNTER) + 1;
     NEW_ST_PT = NEW_ST_PT + 1;
     STATES(NEW_ST_PT) = STADR;
                                                    /*                           */
                                                    /* SET IN NEW VALUE #        */
                                                    /*                           */
     ADDVAL(STADR) = REFNVAL(REFADR);
     IF INADR ¬= 0 THEN
     DO;
          CONCUR = ADDCON(STADR) & "7FFF";
          CONIN = ADDCON(INADR) & "7FFF";
                                                    /*                           */
                                                    /* IF CONSTANT INDICATOR HAS */
```

```
                                                                   /* CHANGED SO HAS STATE      */
                                                                   /*                           */
          IF CONCUR ¬= 0 THEN
          DO;
              IF CONCUR ¬= CONIN THEN
              DO;
                  ADDCON(STADR) = 0;
                  CUR_STATE_CHGD = TRUE;
              END;
          END;
      END;
END BUILD_NEW_SIM_VAR_ENTRY;
                /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
                /*              ASSIGN_REFINEMENT                  */
                /*                                                 */
                /* THIS ROUTINE ASSIGNS A NEW REFINEMENT           */
                /* VALUE TABLE ENTRY.                              */
                /* ROUTINES CALLED:                                */
                /*               ERROR                             */
                /*               GETVAL#                           */
                /*                                                 */
                /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
ASSIGN_REFINEMENT: PROCEDURE(OVAL, IVAL, OTYP,ITYP) FIXED;
      DECLARE (OVAL, IVAL, OTYP, ITYP, TESTA, TESTB) FIXED;
      IF RTOP > MAX_REFINEMENTS THEN
      DO;
          CALL ERROR('REFINEMENT TABLE OVERFLOW', 1);
          RTOP = RTOP - 1;
      END;
      REFCVAL(RTOP) = OVAL;
      REFIVAL(RTOP) = IVAL;
                                                                   /*                           */
                                                                   /* IF CURRENT AND OUTPUT     */
                                                                   /* POOL DO NOT HAVE THE SAME */
                                                                   /* VALUE # ASSIGN A NEW ONE. */
                                                                   /*                           */
      IF (IVAL = OVAL) & (OTYP = ITYP) THEN
      REFNVAL(RTOP) = OVAL;
      ELSE
      REFNVAL(RTOP) = GETVAL#(COMPU);
      RTOP = RTOP + 1;
      RETURN RTOP - 1;
END ASSIGN_REFINEMENT;
                /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
                /*          SEARCH_REFINEMENT_LIST                 */
                /*                                                 */
                /* THIS ROUTINE SEARCHES REFINEMENT LIST           */
                /* FOR AN ENTRY V(OVAL, IVAL) IF FOUND,            */
                /* RETURNS ADDRESS.                                */
                /*                                                 */
                /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
SEARCH_REFINEMENT_LIST: PROCEDURE(OVAL, IVAL) FIXED;
      DECLARE (POINTER, OVAL,IVAL, FOUND) FIXED;
      FOUND = 0;
      POINTER = 1;
                                                                   /*                           */
                                                                   /* SEARCH FOR MATCH IN       */
                                                                   /* REFINEMENT LIST           */
                                                                   /*                           */
      DO WHILE (POINTER < RTOP) & (FOUND = 0);
          IF (REFCVAL(POINTER) = OVAL) & (REFIVAL(POINTER) = IVAL) THEN
          FOUND = POINTER;
          POINTER = POINTER + 1;
      END;
      RETURN FOUND;
END SEARCH_REFINEMENT_LIST;
                /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
                /*                                                 */
                /*               DELETE_OCCURANCE                  */
                /* THIS ROUTINE DECREMENTS THE OCCURANCE           */
                /* COUNT FOR A VALUE #, CHECKS FOR ALL             */
                /* SIMPLE VARIABLE PAU AND DECREMENTS ENTRY        */
                /* COUNT.                                          */
                /* ROUTINES CALLED:                                */
                /*               ERROR                             */
                /*                                                 */
                /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
DELETE_OCCURANCE: PROCEDURE(VALUE#);
      DECLARE(OCCADR, VALUE#, BUBBLE, NEXT, TEMP) FIXED;
      OCCADR = SRCH_OCCURANCE_LIST(VALUE#);
      IF OCCADR = 0 THEN
      DO;
          CALL ERROR('OCCURANCE LIST ERROR',6);
          RETURN;
      END; ELSE
                                                                   /*                           */
                                                                   /* IS CLASS EXHAUSTED        */
                                                                   /*                           */
```

```
     OCCNUM(OCCADR) = OCCNUM(OCCADR) - 1;
     IF OCCNUM(OCCADR) = 0 THEN
     DO;
                                                        /*                            *,
                                                        /* YES...REPACK OCCURANCE     *,
                                                        /* LIST                       *,
                                                        /*                            *,
        BUBBLE = OCCADR;
        DO WHILE BUBBLE < OCCTOP;
           NEXT = BUBBLE + 1;
           OCCNUM(BUBBLE) = OCCNUM(NEXT);
           OCCVAL(BUBBLE) = OCCVAL(NEXT);
           BUBBLE = BUBBLE + 1;
        END;                                            /* BUBBLE < NUM_CLASS         *,
        OCCTOP = OCCTOP - 1;
     END;
     TOT_CLASS_CNT = TOT_CLASS_CNT - 1;
END DELETE_OCCURANCE;
               /*<><><><><><><><><><><><><><><><><><><><><><>*/
               /*                                           */
               /*            PERMUTE_REFINEMENT             */
               /* ROUTINE TO PERMUTE THE REFINEMENT TABLE   */
               /* TO ALLOW COMPLETE CHECKING FOR ALL        */
               /* POSSIBLE INPUT EXPRESSIONS.               */
               /*                                           */
               /*<><><><><><><><><><><><><><><><><><><><><><>*/
PERMUTE_REFINEMENT: PROCEDURE BIT(1);
     DECLARE ANS BIT(1),
             TOP FIXED,
             TEM BIT(16);
     ANS = FALSE;
                                                        /*                            *,
                                                        /* IF FIRST PASS SET POINTERS*,
                                                        /*                            *,
     IF FIRSTCALLR THEN
     DO;
        FIRSTCALLR = FALSE;
        BOPAR = 1;
        BOPBR = 1;
     END;
     TOP = RTOP - 1;
     BOPBR = BOPBR + 1;
                                                        /*                            *,
                                                        /* IF SECOND POINTER BEYOND   *,
                                                        /* TABLE LIMIT ADVANCE FIRST  *,
                                                        /* POINTER AND RESET SECOND   */
                                                        /*                            */
     IF BOPBR > TOP THEN
     DO;
        BOPAR = BOPAR + 1;
        BOPBR = BOPAR + 1;
        IF BOPAR < TOP THEN
        ANS = TRUE;
     END;
     ELSE
     ANS = TRUE;
                                                        /*                            */
                                                        /* IF PERMUTATION WAS POSSIBL*/
                                                        /* PERMUTE THE REFINEMENT     */
                                                        /* LIST.                      */
                                                        /*                            */
     IF ANS THEN
     DO;
        TEM = REFNVAL(BOPBR);
        REFNVAL(BOPBR) = REFNVAL(BOPAR);
        REFNVAL(BOPAR) = TEM;
        TEM = REFCVAL(BOPBR);
        REFCVAL(BOPBR) = REFCVAL(BOPAR);
        REFCVAL(BOPAR) = TEM;
        TEM = REFIVAL(BOPBR);
        REFIVAL(BOPBR) = REFIVAL(BOPAR);
        REFIVAL(BOPAR) = TEM;
     END;
     RETURN ANS;
END PERMUTE_REFINEMENT;
               /*<><><><><><><><><><><><><><><><><><><><><><>*/
               /*                                           */
               /*             SET_SEARCH_LIST               */
               /* THIS ROUTINE SETS UP THE LIST OF OPERAND  */
               /* VALUE NUMBERS TO BE SEARCHED FOR IN THE   */
               /* INPUT POOL.  THE SEARCH LOOKS FOR         */
               /* CORRESPONDING INPUT VALUES IN THE         */
               /* REFINEMENT TABLE FOR THE TWO INPUT VALUE  */
               /* NUMBERS.  EX. (VALA, X) AND (VALB, Y)     */
               /* RESULTING IN     (X, Y)   A VALUE NUMBER  */
               /* PAIR THAT IS EQUIVALENT IN THE INPUT      */
               /* POOL.                                     */
               /*                                           */
```

```
                  /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
SET_SEARCH_LIST:  PROCEDURE BIT(1);
     DECLARE (ANS, TRYFLG, FIND, OVFLOW, COMFLG) BIT(1),
             (SPOINT, CNT, OCNT, OPADR, LSONE) FIXED;
     FIRSTCALL = TRUE;
     COMFLG = TRUE;
     ANS = FALSE;
     STOP = 0;
     SPOINT = 1;
                                                       /*                            */
                                                       /* PROCESS UNTIL ALL          */
                                                       /* PERMUTATIONS IN CURRENT    */
                                                       /* POOL HAVE BEEN SEARCHED    */
                                                       /* FOR.                       */
                                                       /*                            */
     DO WHILE COMFLG;
          TRYFLG = TRUE;
          FIRSTCALLR = TRUE;
          FIND = FALSE;
                                                       /*                            */
                                                       /* PROCESS UNTIL A MATCH IS   */
                                                       /* FOUND OR ALL PERMUTATIONS  */
                                                       /* OF REFINEMENT LIST HAVE    */
                                                       /* BEEN CHECKED.              */
                                                       /*                            */
          DO WHILE TRYFLG & ¬ FIND;
               OVFLOW = FALSE;
               CNT = 1;
               OCNT = 1;
                                                       /*                            */
                                                       /* PROCESS A REFINEMENT LIST  */
                                                       /* CONFIGURATION.             */
                                                       /*                            */
               DO WHILE (¬FIND) & (¬OVFLOW);
                    IF REFCVAL(CNT) = OPERS(OCNT) THEN
                    DO;
                         REFOPERS(SPOINT) = REFNVAL(CNT);
                         INPOPERS(SPOINT) = REFIVAL(CNT);
                         SPOINT = SPOINT + 1;
                         FIND = OCNT = NUMOP;
                         CNT = CNT + 1;
                         OCNT = OCNT + 1;
                         LSONE = OCNT - 1;
                         OVFLOW = (CNT + NUMOP - LSONE) >= RTOP;
                    END;
                    ELSE DO;
                         CNT = CNT + 1;
                         LSONE = OCNT - 1;
                         OVFLOW = (CNT + NUMOP - LSONE) >= RTOP;
                    END;
               END;
               IF OVFLOW THEN
               SPOINT = SPOINT - LSONE;
                                                       /*                            */
                                                       /* IF CURRENT POOL EXPRESSION*/
                                                       /* FOUND, INDICATE SO.        */
                                                       /*                            */
               IF FIND THEN
               DO;
                    ANS = TRUE;
                    STOP = STOP + 1;
               END; ELSE
               TRYFLG = PERMUTE_REFINEMENT;
          END;
                                                       /*                            */
                                                       /* IF CURRENT POOL COMMUNITIV*/
                                                       /* TRY TO PERMUTE.            */
                                                       /*                            */
          IF COMMOP THEN
          DO;
               OPADR = ADDR(OPERS(1));
               COMFLG = PERMUTE(NUMOP, OPADR, 2, TRUE);
          END; ELSE
          COMFLG = FALSE;
     END;
     RETURN ANS;
END SET_SEARCH_LIST;
                  /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                                               */
                  /* ACTUAL BEGINNING OF PERFORM_MEET_OP.          */
                  /*                                               */
                  /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
                                                       /*                            */
                                                       /* SET THE CURRENT STATE      */
                                                       /* COUNTER AND POINTER        */
                                                       /*                            */
     CALL INITIAL_PMO;
     IF TRACETOG THEN
```

```
DO;
    CALL LINE_FEED(1);
    OUTPUT = ' BEFORE THE MEET OPERATION:  ';
    IF PUNCHTOG THEN
    OUTPUT(2)= ' BEFORE THE MEET OPERATION:  ';
    CALL LINE_FEED(1);
    CALL PRINT_OUTPUT_POOL;
    CALL W_STATE(BLKADR, ' CURRENT POOL FOR');
END;
                                                /*                          */
                                                /* IS THERE ENOUGH ROOM TO  */
                                                /* PERFORM THE MEET OPERATION*/
                                                /*                          */
IF (STLOC + STCOUNTER) > MAXSTATE THEN
DO;                                             /* NO...                    */
    CALL ERROR(' UNABLE TO PERFORM MEETOP - STATE TABLE OVERFLOW', 1);
    RETURN FALSE;
END;
                                                /*                          */
                                                /* SET POINTERS TO THE MERGED*/
                                                /* STATE AREA               */
                                                /*                          */
NEW_ST_COUNTER = MAXSTATE - STCOUNTER -1;
NEW_ST_PT = MAXSTATE - STCOUNTER -1;
STATES(NEW_ST_COUNTER) = 0;
                                                /*                          */
                                                /* BUILD THE CLASS #        */
                                                /* OCCURANCE LIST FOR THE   */
                                                /* CURRENT POOL             */
                                                /*                          */
CALL BUILD_CLASS_OCCURANCE_LIST;
IF TABLETOG THEN
CALL W_OCCURRENCE;
                                                /*                          */
                                                /* PROCESS ALL THE SIMPLE   */
                                                /* VARIABLES FROM THE CURRENT*/
                                                /* STATE                    */
                                                /*                          */
                                                /*                          */
                                                /* ARE THERE ANY SIMPLE     */
                                                /* VARIABLES PRESENT.       */
                                                /*                          */
DO WHILE (STCOUNTER > 0) & (SIMP_VARIABLES_PRES ¬= 0);
                                                /*                          */
                                                /* YES...                   */
                                                /*                          */
    STPT = STPT + 1;
    STCOUNTER = STCOUNTER - 1;
    OPTIMADR = STATES(STPT);
    OPTIMVAR = ADDRESS(OPTIMADR);
    OPTIMVAL = ADDVAL(OPTIMADR);
    OPTTYP = ADDTYPE(OPTIMADR);
    IF OPTIMVAR ¬= 0 THEN                       /* IS THIS A SIMPLE VARIABLE */
                                                /* OR AN EXEC STACK ENTRY   */
    DO;
                                                /*                          */
                                                /* YES...IS IT EXEC STACK   */
                                                /*                          */
        IF SHR(OPTIMVAR, 15) & "1" THEN
                                                /*                          */
                                                /* YES...GET THE IDENTIFIER */
                                                /*                          */
        DO;
            OPTIMVAR = ADDCON(OPTIMADR) & "7FFF";
            CURESTACK = TRUE;
        END; ELSE
        CURESTACK = FALSE;
                                                /*                          */
                                                /* DECREMENT THE VALUE #    */
                                                /* COUNT FOR THIS IDENTIFIER */
                                                /*                          */
        CALL DELETE_OCCURANCE(OPTIMVAL);
        SIMP_VARIABLES_PRES = SIMP_VARIABLES_PRES - 1;
                                                /*                          */
                                                /* IS THE IDENTIFIER IN THE */
                                                /* INPUT POOL               */
                                                /*                          */
        INPUTADR = LOOK_VARIABLE(OPTIMVAR);
        IF INPUTADR ¬= 0 THEN
        DO;
                                                /*                          */
                                                /* YES....                  */
                                                /*                          */
            INPTYP = ADDTYPE(INPUTADR);
            INPUTVAL = ADDVAL(INPUTADR);
            REFADR = SEARCH_REFINEMENT_LIST(OPTIMVAL, INPUTVAL);
                                                /*                          */
                                                /* IS THE ENTRY ALREADY IN  */
```

```
                                                    /* THE REFINEMENT LIST        */
                                                    /*                            */
          IF REFADR = 0 THEN
                                                    /*                            */
                                                    /* NO...PUT IT IN             */
                                                    /*                            */
          REFADR = ASSIGN_REFINEMENT(OPTIMVAL, INPUTVAL, OPTTYP, INPTYP);
                                                    /*                            */
                                                    /* BUILD A STATE ENTRY FOR    */
                                                    /* THE SIMPLE VARIABLE        */
                                                    /*                            */
          CALL BUILD_NEW_SIM_VAR_ENTRY(REFADR, OPTIMADR, INPUTADR);
                                                    /*                            */
       END; ELSE                                    /* END OF INPUTADR ¬= 0       */
                                                    /*                            */
                                                    /* VARIABLE NOT IN INPUT POOL */
                                                    /* I.E., CURRENT STATE IS     */
                                                    /* GOING TO GET SMALLER       */
                                                    /*                            */
       CUR_STATE_CHGD = TRUE;
    END;                                            /* END OF SIMPLE VARIABLE     */
END;
IF TABLETOG THEN
DO;
    OUTPUT = ' AFTER PROCESSING SIMPLE VARIABLES:';
    IF PUNCHTOG THEN
    OUTPUT(2) = ' AFTER PROCESSING SIMPLE VARIABLES:';
    CALL W_OCCURRENCE;
    CALL W_REFINEMENT;
END;
                                                    /*                            */
                                                    /* NOW PROCESS ALL EXPRESSION */
                                                    /* FROM CURRENT STATE         */
                                                    /*                            */
DO WHILE TOT_CLASS_CNT > 0;
    STCOUNTER = SAVCNT;
    STPT = SAVST;
                                                    /*                            */
                                                    /* PROCESSES ALL CLASSES IN   */
                                                    /* CURRENT POOL.              */
                                                    /*                            */
    DO WHILE STCOUNTER > 0;
       STPT = STPT + 1;
       STCOUNTER = STCOUNTER - 1;
       OPTIMADR = STATES(STPT);
                                                    /*                            */
                                                    /* IS CLASS PRESENT           */
                                                    /*                            */
       IF OPTIMADR ¬= 0 THEN
       DO;
                                                    /*                            */
                                                    /* YES...SET INDICATOR        */
                                                    /*                            */
          OPTIMVAR = ADDRESS(OPTIMADR);
          OPTIMVAL = ADDVAL(OPTIMADR);
          OPTTYP = ADDTYPE(OPTIMADR);
                                                    /*                            */
                                                    /* IS THIS AN EXPRESSION      */
                                                    /* ENTRY                      */
                                                    /*                            */
          IF OPTIMVAR = 0 THEN
          DO;
                                                    /*                            */
                                                    /* YES...                     */
                                                    /*                            */
             EXPADR = ADDNAM(OPTIMADR);
             OPER, TEMP = OPTTYP;
             COMMOP = OP_TYPE(OPER) = COMM;
             NUMOP = OP_DEGL(OPER);
             FOUND = FALSE;
                                                    /*                            */
                                                    /* CHECK TO SEE THAT ALL THE  */
                                                    /* OPERANDS HAVE BEEN REMOVED */
                                                    /* FROM THE CLASS OCCURANCE   */
                                                    /* LIST                       */
                                                    /*                            */
             I = 1;
             DO WHILE NUMOP > 0 & ¬ FOUND;
                OPVAL = VTAB(EXPADR);
                OPERS(I) = OPVAL;
                OCCADR = SRCH_OCCURANCE_LIST(OPVAL);
                IF OCCADR ¬= 0 THEN
                IF OPVAL ¬= OPTIMVAL THEN
                FOUND = TRUE;
                I = I + 1;
                EXPADR = EXPADR - 1;
                NUMOP = NUMOP - 1;
             END;                                   /* WAS OPERAND CHECK OK       */
```

```
NUMOP = OP_DEGL(OPER);
IF ¬FOUND THEN
DO;
                                        /*                           */
                                        /* YES...PROCESS THE EXPRESSN*/
                                        /*                           */
    CALL DELETE_OCCURANCE(OPTIMVAL);
    EXPADR = ADDNAM(OPTIMADR);
    CURES = OPTIMVAL;
                                        /*                           */
                                        /* ARE THERE ANY EQUIVALENCE */
                                        /* CLASSES TO TEST FOR       */
                                        /*                           */
    IF SET_SEARCH_LIST THEN
    DO;
                                        /*                           */
                                        /* YES...CHECK ALL EXPRESSION*/
                                        /* TABLE ENTRIES VIA THE HASH*/
                                        /* CODED CHAIN.              */
                                        /*                           */
        SPOINT = 1;
            SVPNT = SPOINT;
        DO WHILE (STOP > 0) & (¬ FOUND);
            HASHADR = OPER;
            DO WHILE NUMOP > 0;
                HASHADR = INPOPERS(SPOINT) + HASHADR;
                SPOINT = SPOINT + 1;
                NUMOP = NUMOP - 1;
            END;
            NUMOP = OP_DEGL(OPER);
            SPOINT = SVPNT;
            HASHADR = (HASHADR) MOD HASHBASE;
            HASHADR = VTAB(HASHADR);
            DO WHILE (HASHADR ¬= 0) & (¬ FOUND);
                INPUTADR = 0;
                I = ADDLOC;
                DO WHILE (I > 0) & (INPUTADR = 0);
                    IF ADDRESS(I) = 0 THEN
                    IF ADDNAM(I) = HASHADR THEN
                    INPUTADR = I;
                    I = I - 1;
                END;
                IF INPUTADR ¬= 0 THEN
                DO;
                TSTOPER = VTAB(HASHADR);
                TSTOPER = TSTOPER & "FF";
                HASHADR = HASHADR + 2;
                IF TSTOPER = OPER THEN
                DO;
                    FIND = TRUE;
                    I = 0;
                    DO WHILE NUMOP > 0;
                        IF VTAB(HASHADR + I) ¬= INPOPERS
                           (SPOINT) THEN
                        FIND = FALSE;
                        I = I + 1;
                        SPOINT = SPOINT + 1;
                        NUMOP = NUMOP - 1;
                    END;
                    NUMOP = OP_DEGL(OPER);
                    SPOINT = SVPNT;
                    IF FIND THEN
                    DO;
                                        /*                           */
                                        /* AN INPUT EXPRESSION HAS   */
                                        /* BEEN MATCHED.             */
                                        /*                           */
                        INPTYP = OPTTYP;
                        INRES = VTAB(HASHADR + NUMOP);
                        REFADR = SEARCH_REFINEMENT_LIST
                              (CURES, INRES);
                        IF REFADR = 0 THEN
                        REFADR = ASSIGN_REFINEMENT(CURES,
                              INRES, OPTTYP, INPTYP);
                        CALL BUILD_EXP_ENTRY(OPTIMADR,
                              REFADR, INPUTADR);
                        FOUND = TRUE;
                    END;
                END;
                                        /*                           */
                                        /* ADCANCE TO NEXT HASH CHAIN*/
                                        /* ENTRY.                    */
                                        /*                           */
                HASHADR = HASHADR - 1;
                END;
                ELSE
                HASHADR = HASHADR + 1;
                HASHADR = VTAB(HASHADR);
```

```
                                        END;                   /* HASH CODE CHAIN              */
                                                               /*                              */
                                                               /* ADVANCE TO NEXT INPUT        */
                                                               /* EXPRESSION TO SEARCH FOR     */
                                                               /*                              */
                                        NUMOP = OP_DEGL(OPER);
                                        SVPNT = NUMOP + SVPNT;
                                        SPOINT = SVPNT;
                                        STOP = STOP - 1;
                                    END;
                                END;
                                                               /*                              */
                                                               /* CLEAR CLASS INDICATOR TO     */
                                                               /* PRECLUDE DUPLICATE           */
                                                               /* PROCESSING.                  */
                                                               /*                              */
                                STATES(STPT) = 0;
                            END;
                                                               /*                              */
                                                               /* IF EXPRESSION NOT FOUND      */
                                                               /* STATE HAS CHANGED.           */
                                                               /*                              */
                            IF ¬FOUND THEN
                            CUR_STATE_CHGD = TRUE;
                        END;
                    END;
                END;
            END;
                                                               /*                              */
                                                               /* MOVE THE NEW STATE INTO      */
                                                               /* THE PROPER LOCATIONS         */
                                                               /*                              */
            STCOUNTER = STATES(NEW_ST_COUNTER);
            STPT = SAVST;
            STATES(STPT) = STCOUNTER;
            NEW_ST_PT = NEW_ST_COUNTER;
            DO WHILE STCOUNTER > 0;
                STPT = STPT + 1;
                NEW_ST_PT = NEW_ST_PT + 1;
                STCOUNTER = STCOUNTER - 1;
                STATES(STPT) = STATES(NEW_ST_PT);
            END;
            IF TABLETOG THEN
            DO;
                OUTPUT = ' AFTER PROCESSING ALL EXPRESSIONS:';
                IF PUNCHTOG THEN
                OUTPUT(2) = ' AFTER PROCESSING ALL EXPRESSIONS:';
                CALL W_OCCURRENCE;
                CALL W_REFINEMENT;
            END;
            IF TRACETOG THEN
            DO;
                OUTPUT = ' AFTER PERFORMING THE MEET OPERATION:  ';
                IF PUNCHTOG THEN
                OUTPUT(2) = ' AFTER PERFORMING THE MEET OPERATION:  ';
                IF ¬ CUR_STATE_CHGD THEN
                DO;
                    OUTPUT = ' NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT PO
        OL';
                    IF PUNCHTOG THEN
                    OUTPUT(2) =' NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT
        POOL';
                END;
                ELSE
                CALL W_STATE(BLKADR,' NEW CURRENT POOL/INPUT POOL FOR ');
                CALL LINE_FEED(1);
            END;
            RETURN CUR_STATE_CHGD;
        END PERFORM_MEET_OP;
                        /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
                        /*                                               */
                        /*                 BUILD_BLOCK                   */
                        /* BUILDS THE BLOCK HEADER AND CLEARS THE        */
                        /* LOCATIONS FOR BRANCH POINTS                   */
                        /*        BLOCK# | PROCESSED BIT | STATE PTR     */
                        /*        FWA           | LWA                    */
                        /*        # OF TRANSVERSALS                      */
                        /*        # OF REFERENCES                        */
                        /*        BRANCH FROM                            */
                        /*        BRANCH FROM                            */
                        /* L IS THE FWA ADDRESS IN CODE OF BLOCK TO      */
                        /* BE BUILT.                                     */
                        /* ROUTINES CALLED:                              */
                        /*              ERROR                            */
                        /*              BUILD_INPUT_POOL                 */
                        /*              W_STATE                          */
                        /*                                               */
                        /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
```

```
BUILD_BLOCK: PROCEDURE(L) FIXED;
     DECLARE (L,C,I,J,B,S) FIXED;
     C = CSFCODE(L);
     I = C & "7FFF";
                                                                 /*                          */
                                                                 /* ADVANCE BLK COUNTER      */
                                                                 /*                          */
     BLKCNT = BLKCNT + 1;
     J = BLKLOC + I + BLKSIZE;                                   /* GET LWA BLK HEADER ADDRESS*/
     IF J > MAXCODE THEN
     DO;
          CALL ERROR(' BASIC BLOCK TABLE OVERFLOW', 1);
          BLKLOC = CODELOC + 1;
     END;
                                                                 /*                          */
                                                                 /* SAVE THE BLOCK HEADER    */
                                                                 /* ADDRESS FOR LATER USE.   */
                                                                 /*                          */
     B = BLKLOC;
     BLKPNT(BLKCNT) = B;
                                                                 /*                          */
                                                                 /* BUILD INPUT POOL FROM    */
                                                                 /* IMMEDIATE PREDECESSOR BLK */
                                                                 /*                          */
     S = BUILD_INPUT_POOL;
                                                                 /*                          */
                                                                 /* SET FWA OF BLOCK HEADER  */
                                                                 /* IN CODE AND MARK BLOCK AS */
                                                                 /* ENTERED                  */
                                                                 /*                          */
     CSFCODE(L)=(C&"FFFF0000") | SHL(1,15) | B;
     CSFCODE(B) = SHL(BLKCNT, 16) | S;                           /* BLK# | STATE POINTER     */
     IF TRACETOG THEN
     CALL W_STATE(B, ' INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR ');
     CSFCODE(B + 1) = SHL(L, 16);                                /* START | FINISH           */
     CSFCODE(B + 2) = 0;                                         /* # OF TRAVERSALS          */
     CSFCODE(B + 3) = I;                                         /* NUMBER OF REFERENCES FOR */
                                                                 /* THIS BLOCK.              */
     BLKLOC=J;
     J=B+4;
                                                                 /*                          */
                                                                 /* CLEAR LOCATIONS INTO WHICH*/
                                                                 /* PREDECESSOR BLOCK #'S WILL*/
                                                                 /* BE PLACED LATER          */
                                                                 /*                          */
     DO WHILE I > 0;
          CSFCODE(J) = 0;
          J = J + 1;
          I = I - 1;
     END;
END BUILD_BLOCK;
                /*<><><><><><><><><><><><><><><><><><><><><><>*/
                /*                                            */
                /*              RESTORE_TOGGLES               */
                /* SETS TOGGLES AS SPECIFIED BY A PARTICULAR*/
                /* BLOCK                                      */
                /*                                            */
                /*<><><><><><><><><><><><><><><><><><><><><><>*/
RESTORE_TOGGLES: PROCEDURE;
     DECLARE (I,J) FIXED;
     I=SAV_TOG(CTOP);
     DO J = 0 TO #TOGS;
          TOGGLES(J) = SHR(I, J) & "1";
     END;
END RESTORE_TOGGLES;
                /*<><><><><><><><><><><><><><><><><><><><><><>*/
                /*                                            */
                /*                  CONTROL_FLOW              */
                /* THIS IS THE BASIC CONTROL LOOP OF THE      */
                /* OPTIMIZATION PROCESS.  USING THE LIST      */
                /* OF BLOCKS TO BE PROCESSED, A OPTIMIZED     */
                /* BLOCK IS SELECTED AND ITS CURRENT          */
                /* OPTIMIZED STATE (INPUT POOL; S.N) IS       */
                /* RETRIEVED.  BASIC BLOCK THEN FORMS         */
                /* F(N, S.N).  THIS IS THEN SET AS THE NEW    */
                /* INPUT POOL FOR ALL SUCCESSORS OF THE       */
                /* BLOCK JUST PROCESSED.  AT THIS POINT,      */
                /* THE "MEET" OPERATOR IS ACCOMPLISHED.       */
                /* CONTROL TABLE IS:                          */
                /*   BIT31        30-16        15-0           */
                /*   VALID      PREDECESSOR FWA OF BLK IN     */
                /*   ENTRY        BLOCK#      CODE AREA       */
                /*                                            */
                /* THE VALID BIT, IF SET IS AN INDICATOR TO */
                /* PRECLUDE THE PROCESSING OF A BLOCK ON      */
                /* THE CONTROL STACK.                         */
                /* ROUTINES CALLED:                           */
                /*              RESTORE_TOGGLES               */
```

```
          /*               SET_UP_BLOCK                */
          /*               DECODE                      */
          /*               W_ALL                       */
          /*               BASIC_BLOCK                 */
          /*               ENTERADD                    */
          /*               BUILD_INPUT_POOL            */
          /*               PERFORM_MEET_OP             */
          /*               BUILD_BLOCK                 */
          /*               ORDER_BLOCK_LIST            */
          /*                                           */
          /*<><><><><><><><><><><><><><><><><><><><><>*/
CONTROL_FLOW: PROCEDURE;
    DECLARE (C, W, B, NBLK#, I, K, S) FIXED;
    DO WHILE CTOP > 0;
                                                        /*                             */
                                                        /* GET BLOCK TO BE PROCESSED */
                                                        /* OFF LIST                    */
                                                        /*                             */
        C = CONTROL(CTOP);
        CALL RESTORE_TOGGLES;
        ALTCT = 1;
                                                        /*                             */
                                                        /* IS BLOCK TO BE PROCESSED   */
                                                        /*                             */
        IF SHR(C, 31) THEN                              /* NO...BYPASS IT              */
        CTOP = CTOP - 1;
        ELSE
        DO;
                                                        /*                             */
                                                        /* YES...SET FWA THIS BLOCK   */
                                                        /* AND BLK# COUNTER.          */
                                                        /*                             */
            M_CONTROL = C & "FFFF";
            BLK# = SHR(C, 16);
                                                        /*                             */
                                                        /* BLOCK HEADER ADDRESS        */
                                                        /*                             */
            W = CSFCODE(M_CONTROL);
            B, CURBLOCK = W & "7FFF";
                                                        /*                             */
                                                        /* SET UP TABLES ACCORDING TO*/
                                                        /* THE CURRENT STATE OF THIS */
                                                        /* BLOCK                       */
                                                        /*                             */
            CALL SET_UP_BLOCK;
            CTOP = CTOP + 1;
            NBLK# = SHR(CSFCODE(B), 16);
                                                        /*                             */
                                                        /* ADVANCE THE PASS COUNTER   */
                                                        /*                             */
            PASS#, CSFCODE(B + 2) = CSFCODE(B + 2) + 1;
            IF TRACETOG THEN
            DO;
                OUTPUT = '  CONTROL AT ' || M_CONTROL || ', FROM BLK ' || BLK#||
                         ' TO BLK ' || NBLK# || ' (PASS ' || PASS# ||')';
                IF PUNCHTOG THEN
                OUTPUT(2)=' CONTROL AT ' || M_CONTROL || ', FROM BLK ' || BLK#||
                         ' TO BLK ' || NBLK# || ' (PASS ' || PASS# ||')';
                CALL DECODE(M_CONTROL, W, 'C+');
            END;
            ALTST = 1;
            IF TABLETOG THEN
            CALL W_ALL;
            B = B + 3;
            I = CSFCODE(B);
            K = 1;
                                                        /*                             */
                                                        /* SET IN THE PREDECESSOR     */
                                                        /* BLOCK # FOR THIS BLOCK     */
                                                        /*                             */
            DO WHILE K <= I;
                C = CSFCODE(B + K);
                IF C = 0 THEN
                DO;
                    CSFCODE(B + K) = BLK#;
                    K = I;
                END;
                IF C = BLK# THEN
                K = I;
                K = K + 1;
            END;
                                                        /*                             */
                                                        /* SET CURRENT BLK# TO BE     */
                                                        /* NEW BLOCK                   */
                                                        /*                             */
            BLK# = NBLK#;
                                                        /*                             */
                                                        /* GO PROCESS THE BLOCK        */
```

```
                                        /*                            */
CALL BASIC_BLOCK;
                                        /*                            */
                                        /* COPY EXEC STACK TO ADDRESS*/
                                        /* STACK BEFORE PROCESSING    */
                                        /*                            */
K = ETOP;
DO WHILE ETOP > 0;
    CALL ENTERADD(SHL(1, 15) | (K - ETOP));
    ADDTYPE(ADDLOC) = EXTYPE(ETOP);
    ADDVAL(ADDLOC) = EXVAL#(ETOP);
    ADDNAM(ADDLOC) = EXNAM#(ETOP);
    I = EXCON(ETOP);
    ADDCON(ADDLOC) = SHL(I, 15) | EXADD(ETOP);
    ETOP = ETOP - 1;
END;
                                        /*                            */
                                        /* COPY VALUE TABLE ENTRIES   */
                                        /* TO ADDRESS STACK.          */
                                        /* - ADDRESS FIELD SET TO 0   */
                                        /* TO INDICATE VTAB ENTRY     */
                                        /* ADDNAM POINTS TO VTAB      */
                                        /* ENTRY.                     */
                                        /*                            */
K = VTAB(VTOP);
DO WHILE K ¬= 0;
    C = VTAB(K + 1) & "FF";
    W = OP_DEGL(C) + K + 3;
    W = VTAB(W);
    DO;
        CALL ENTERADD(0);
        ADDTYPE(ADDLOC) = C;
        ADDVAL(ADDLOC) = W;
        ADDNAM(ADDLOC) = K + 1;
        ADDCON(ADDLOC) = 0;
    END;
    K = VTAB(K);
END;
                                        /*                            */
                                        /* IF FIRST PASS FOR THIS     */
                                        /* BLOCK, SET IN THE LWA IN   */
                                        /* BLOCK CONTROL AREA.        */
                                        /*                            */
IF PASS# = 1 THEN
CSFCODE(CURBLOCK + 1) = (CSFCODE(CURBLOCK + 1) & "FFFF0000") |
        M_CONTROL;
                                        /*                            */
                                        /* PERFORM THE MEET OPERATION*/
                                        /* ON THE NEW BLOCKS ENTERED */
                                        /* ON THE CONTROL STACK.  IF */
                                        /* THE SAME BLOCK IS ON THE   */
                                        /* CONTROL STACK MORE THAN    */
                                        /* ONCE, MARK ALL BUT THE ONE*/
                                        /* WITH THE MINIMUM CURRENT   */
                                        /* POOL TO BE BYPASSED.  IF   */
                                        /* THE MEET OPERATION DOES    */
                                        /* NOT DECREASE THE CURRENT   */
                                        /* OPTIMIZED POOL OF A BLOCK */
                                        /* MARK IT NOT TO BE PROCESSD*/
                                        /*                            */
K = #XITS;
K = CTOP - K;
DO WHILE K < CTOP;
    K = K + 1;
                                        /*                            */
                                        /*                            */
                                        /* GET BLOCK CODE FWA         */
    I = CONTROL(K);
    I = I & "FFFF";
    W = CSFCODE(I);
                                        /*                            */
                                        /* GET BLOCK HEADER FWA       */
                                        /*                            */
    C = W & "7FFF";
    IF SHR(W, 15) & "1" THEN            /* IF BLOCK IS PRESENT        */
    DO;                                 /* PERFORM THE "MEET" OPER    */
        IF PERFORM_MEET_OP(C) THEN
                                        /*                            */
                                        /* OPTIMIZED POOL WAS ALTERED*/
                                        /* - IE., THIS VERSION OF     */
                                        /* THIS BLOCK IS MOST CURRENT*/
                                        /*                            */
        DO B = 1 TO K - 1;
                                        /*                            */
                                        /* IS SAME BLOCK ON STACK     */
                                        /* MORE THAN ONCE.            */
                                        /*                            */
            IF (CONTROL(B) & "FFFF") = I THEN
```

```
                                                               /*                           */
                                                               /* YES...MARK EARLIER ENTRIES*/
                                                               /* TO PREVENT PROCESSING     */
                                                               /*                           */
                          CONTROL(B) = SHL(1, 31);
                      END; ELSE
                                                               /*                           */
                                                               /* OPTIMIZED POOL WAS NOT    */
                                                               /* ALTERED DURING MEET - DO  */
                                                               /* NOT PROCESS THIS BLOCK    */
                                                               /*                           */
                      CONTROL(K) = SHL(1, 31);
                  END; ELSE
                                                               /*                           */
                                                               /* BLOCK NOT BUILT      -    */
                                                               /* BUILD A HEADER FOR IT     */
                                                               /* FWA IS I IN CODE          */
                                                               /*                           */
                  CALL BUILD_BLOCK(I);
              END;
                                                               /*                           */
                                                               /* WHILE K < CTOP            */
                                                               /*                           */
                                                               /* ORDER THE BLOCK SELECTION */
                                                               /* LIST AS REQUESTED.        */
              CALL ORDER_BLOCK_LIST;
              IF TABLETOG THEN
              CALL W_ALL;
                                                               /*                           */
                                                               /* END OF BLOCK PROCESSED    */
                                                               /*                           */
          END;
      END;                                                     /* WHILE CTOP > 0            */
END CONTROL_FLOW;
              /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
              /*                                              */
              /*                 INITIAL_CSF                  */
              /* ROUTINE IS THE BASIC INITIALIZATION          */
              /* LOOP FOR THE PROGRAM.                        */
              /*                                              */
              /*<><><><><><><><><><><><><><><><><><><><><><><><>*/
INITIAL_CSF: PROCEDURE;
    DECLARE I FIXED;
    BLANKS='                ';
                                                               /*                           */
                                                               /* SET BASIC OPERATIONS TYPES*/
                                                               /*                           */
    COLLEC=1;
    LOAD = 2;
    STORE = 3;
    STORET = 4;
    DELETE = 5;
    DUPLIC = 6;
    CONV = 7;
    COMM = 8;
    EXCH = 9;
    COND = 10;
    UCOND = 11;
    DO I = 0 TO NTYPES;
        NULL(I) = I;
    END;
    DO I = 1 TO NOPCODES;
        REFER(I - 1) = I + NTYPES;
    END;
    GOSUB = 12;
    RETSUB = 13;
    ETOP = 0;
    VTOP = HASHBASE;
                                                               /*                           */
                                                               /* CLEAR HASH ADDRESSES      */
                                                               /*                           */
    DO I = 0 TO HASHBASE;
        VTAB (I) = 0;
    END;
                                                               /*                           */
                                                               /* SET TOGGLES               */
                                                               /*                           */
    TRACETOG = 0;
    TABLETOG = 0;
    PROPTOG = 1;
    BRANCHTOG = 1;
    PUNCHTOG = 0;
                                                               /*                           */
                                                               /* SET ALL TABLE POINTERS    */
                                                               /*                           */
    CONLOC, ADDLOC, STLOC = 0;
    ADDLOCR = MAXADD + 1;
    ETOPR = MAXEXEC+1;
```

```
      VTOPR=MAXVAL+1;
                                                                        /*                           */
                                                                        /* CLR PRINT TABLES INDICATOR*/
                                                                        /*                           */
      ALTST = 0;
      ALTEX = 0;
      ALTAD = 0;
      ALTCN = 0;
      ALTCT = 0;
      ALTVL = 0;
      PAGE_EJECT;
      OUTPUT = '                           ***CODE SYNTHESIS FILTER***';
      IF PUNCHTOG THEN
      OUTPUT (2) = '                       ***CODE SYNTHESIS FILTER***';
      CALL LINE_FEED(2);
END INITIAL_CSF;
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                                            */
                  /*              INITIAL_METAEX                */
                  /* SETS UP TO PERFORM THE META EXECUTION      */
                  /* ROUTINES CALLED:                           */
                  /*              READ_SELECT_METHOD            */
                  /*              ERROR                         */
                  /*              LINE_FEED                     */
                  /*              READ_CODE                     */
                  /*              FORMATCODEDUMP                */
                  /*              BUILD_CONS_TABLE              */
                  /*              PROPAGATE                     */
                  /*              W_CONS                        */
                  /*              INSRT_SUC_BLOCK               */
                  /*              BUILD_BLOCK                   */
                  /*                                            */
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
INITIAL_METAEX: PROCEDURE;
      DECLARE I FIXED, T CHARACTER;
                                                                        /*                           */
                                                                        /* INPUT THE INTERMEDIATE    */
                                                                        /* LANGUAGE.                 */
                                                                        /*                           */
      SELECT_FLAG = READ_SELECT_METHOD;
      IF SELECT_FLAG > NUM_SELECT_METHODS THEN
      DO;
          CALL ERROR('ILLEGAL SELECTION METHOD CODE - LIFO FORCED'     , 5);
          SELECT_FLAG = 0;
      END;
      T = ' BLOCK SELECTION METHOD WILL BE ';
      T = T || SELECT_METHOD(SELECT_FLAG);
      OUTPUT = T;
      IF PUNCHTOG THEN
      OUTPUT(2) = T;
      CALL LINE_FEED(2);
      CALL READ_CODE;
      IF TRACETOG THEN
      CALL FORMATCODEDUMP;
                                                                        /*                           */
                                                                        /* BUILD THE CONSTANT TABLE  */
                                                                        /*                           */
      CALL BUILD_CONS_TABLE;
      ETOP=1;
                                                                        /*                           */
                                                                        /* ENTER CONSTANTS BY TYPES  */
                                                                        /*                           */
      DO I = 1 TO CONLOC;
          EXADD(ETOP) = I;
          CALL PROPAGATE(CONTYPE(I),0);
      END;
      ETOP=0;
      IF TRACETOG THEN
      CALL W_CONS;
                                                                        /*                           */
                                                                        /* FORCE ENT AS FIRST INSTR  */
                                                                        /* IN THE INTERMEDIATE CODE  */
                                                                        /*                           */
      CSFCODE(#PARMS) = SHL(ENT, 16);
      M_CONTROL = #PARMS;
                                                                        /*                           */
                                                                        /* ENTER BLKO ON CNTROL STACK*/
                                                                        /*                           */
      BLKLOC=CODELOC+1;
      BLKCNT = 0;
      BLK# = 0;
      OPTOP = 1;
      CTOP = 0;
      CALL INSRT_SUC_BLOCK(#PARMS);
      #XITS = 0;
                                                                        /*                           */
                                                                        /* BUILD BLKO HEADER         */
                                                                        /*                           */
      CALL BUILD_BLOCK(M_CONTROL);
```

```
     CALL BUILD_BLOCKTM_CONTROL;
                                                                /*                            */
                                                                /* INPUT THE SELECT METHOD    */
                                                                /* INDICATOR.                 */
                                                                /*                            */
END INITIAL_METAEX;
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                                            */
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                                            */
                  /*                MAIN PROGRAM                */
                  /* THIS IS THE MAIN CONTROL DRIVER FOR THE    */
                  /* ENTIRE CSF PROGRAM.                        */
                  /* ROUTINES CALLED:                           */
                  /*                INITIAL_CSF                 */
                  /*                INITIAL_METAEX              */
                  /*                DUMP_ALL                    */
                  /*                CONTROL_FLOW                */
                  /*                OUTPUT_OPTIMIZATION         */
                  /*                LINE_FEED                   */
                  /*                                            */
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                                            */
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                                                                /*                            */
                                                                /* INITIALIZE POINTERS AND    */
                                                                /* TABLES                     */
                                                                /*                            */
     CALL INITIAL_CSF;
                                                                /*                            */
                                                                /* SET UP FOR META EXECUTION  */
                                                                /*                            */
     CALL INITIAL_METAEX;
     CALL DUMP_ALL;
                                                                /* SAVE THE START TIME OF     */
                                                                /* OPTIMIZATION               */
                                                                /*                            */
     STRTIME = TIME;
                                                                /*                            */
                                                                /* BASIC PROCESS ROUTINE -    */
                                                                /* RETURNS WHEN ALL           */
                                                                /* OPTIMIZATION IS COMPLETE   */
                                                                /*                            */
     CALL CONTROL_FLOW;
                                                                /*                            */
                                                                /* OUTPUT THE DETECTED        */
                                                                /* OPTIMIZATIONS.             */
                                                                /*                            */
     CALL OUTPUT_OPTIMIZATION;
                                                                /* SAVE THE FINISH TIME OF    */
                                                                /* OPTIMIZATION               */
                                                                /*                            */
     CALL LINE_FEED(1);
     PAUTIME = TIME;
     TOTIME = PAUTIME - STRTIME;
     SECONDS = TOTIME / 1000;
     PRTSEC = TOTIME MOD 1000;
     OUTPUT = ' TIME FOR THE OPTIMIZATION WAS ' || SECONDS || '.' || PRTSEC ||
            ' SECONDS.';
     IF PUNCHTOG THEN
     OUTPUT(2)=' TIME FOR THE OPTIMIZATION WAS ' || SECONDS || '.' || PRTSEC ||
            ' SECONDS.';
     CALL LINE_FEED(2);
     OUTPUT = '                    ***CODE SYNTHESIS FILTER IS COMPLETE***';
     IF PUNCHTOG THEN
     OUTPUT(2) = '                 ***CODE SYNTHESIS FILTER IS COMPLETE***';
     PAGE_EJECT;
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                  /*                   FINI                     */
                  /*          CODE SYNTHESIS FILTER             */
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
                  /*     END        END        END        END   */
                  /*<><><><><><><><><><><><><><><><><><><><><><><>*/
EOF EOF EOF
EOF EOF EOF
```

BIBLIOGRAPHY

1. Aho, A. V., Sethi, R. and Ullman, J. D., "Code Optimization on Finite Church-Rosser Systems," _Design and Optimization of Compilers_, p. 69-106, 1972.

2. Aho, A. V. and Ullman, J. D., "Optimization of Straight Line Code," _SIAM Journal on Computing_, v. 1, n. 1, p. 1-19, March 1972.

3. Aho, A. V. and Ullman, J. D., _The Theory of Parsing, Translation, and Compiling Vol. II: Parsing_, Prentice-hall, Inc., 1973.

4. Allen, F. E. and Cocke, J., "A Catalogue of Optimizing Transformations," _Design and Optimization of Compilers_, p. 1-30, 1972.

5. Anderson, J., "A Note on some Compiling Algorithms," _Communications of the ACM_, v. 7, n. 3, p. 149-150, March 1964.

6. Bagwell Jr., J. T., "Local Optimizations," _SIGPLAN Notices of the ACM_, v. 5, n. 7, p. 52-66, July 1970.

7. Busam, V. and Englund, D,, "Optimization of Expressions in FORTRAN," _Communications of the ACM_, v. 12, n. 12, p. 666-674, December 1969.

8. Cheatham, T. E. and Standish T., "Optimization Aspects of Compiler-Compilers," _SIGPLAN Notices of the ACM_, v. 5, n. 10, p. 10-17, October 1970.

9. Cocke, J., "Global Common Subexpression Elimination," _SIGPLAN Notices of the ACM_, v. 5, n. 7, p. 20-24, july 1970.

10. Cocke, J. and Schartz, J. T., _Programming Languages and their Compilers, Preliminary Notes_, 2nd ed., p. 306-524, Courant Institute of Mathematical Sciences New York University, 1970.

11. Day, W., "Compiler Assignment of Data Items to Registers ," _IBM Systems Journal_, v. 9, n. 4, p. 281-317, 1970.

12. Earnest, C. P., "Some Topics in Code Optimization," _Journal of the ACM_, v. 21, n. 1, p. 76-102, January 1974.

13. Elson, M., and Rake, S., "Code Generation Technique for Large Language Compilers," _IBM Systems Journal_, v. 9, n. 3, p. 166-188, 1970.

14. Fateman, R., "Optimal Code for Serial and Parallel Computation," _Communications of the ACM_, v. 12, n. 12, p. 694-695, December 1969.

15. Finkelstein, M., "A Compiler Optimization Technique," _Computer Review_, p. 22-25, February 1968.

16. Floyd, R., "An Algorithm for Coding Efficient Arithmetic Operations," _Communications of the ACM_, v. 4, n. 1, p. 42-51, January 1961.

17. Fong, A., Kam, J. and Ullman, J., "Application of Lattice Algebra to Loop Optimization," _Conference Record of the Second ACM Symposium on Principles of Programming Languages_ , p. 1-9, Palo Alto, California, January 1975.

18. Frailey, D. J., "Expression Optimization Using Unary Complement Operators," _SIGPLAN Notices of the ACM_, v. 5, n. 7, p. 67-85, July 1970.

19. Goldberg, P., "A Comparison of Certain Optimization Techniques," Design and Optimization of Compilers, p. 31-50, 1972.

20. Graham, S. L. and Wegman, M., "A Fast and Usually Linear Algorithm for Global Flow Analysis," Conference Record of the Second ACM Symposium on Principles of Programming Languages, p. 22-34, Palo Alto, California, January 1975.

21. Haynes, H. R. and Schutte, L. J., "Compilation of Optimized Syntactic Recognizers from Floyd-Evans Productions," SIGPLAN Notices of the ACM, v. 5, n. 7, p. 38-51, July 1970.

22. Hecht, M. S. and Ullman, J. D., "Analysis of a Simple Algorithm for Global Data Flow Problems," Conference Record of the ACM Symposium on Principles of Programming Languages, p. 207-217, Boston, Massachusettes, October 1973.

23. Hecht, M. S. and Ullman, J. D., "Characterizations of Reducible Flow Graphs," Journal of the ACM, v. 21, n. 3, p. 367-375, July 1974.

24. Hecht, M. S. and Ullman, J. D., "Flow Graph Reducibility," SIAM Journal on Computing, v. 1, n. 2, p. 188-202, June 1972.

25. Kennedy, K., "A Global Flow Analysis Algorithm," International Journal of Computer Mathematics, Section A, v. 3, n. 1, p. 5-15, December 1971.

26. Kennedy, K., "Node Listings Applied to Data Flow Analysis," Conference Record of the Second ACM Symposium on Principles of Programming Languages, p. 10-21, Palo Alto, California, January 1975.

27. Kennedy, K., "Safety of Code Motion," International Journal of Computer Mathematics, Section A, v. 3, n. 2, p. 117-130, September 1972.

28. Kildall, G. A., "A Unified Approach to Global Optimization," Conference Record of the ACM Symposium on Principles of Programming Languages, p. 194-206, Boston, Massachusettes, October 1973.

29. Kildall, G. A, A Code Synthesis "Filter" for Basic Block Optimization, Computer Science Group Report TR #72-3-01, University of Washington, Seattle, January 1972.

30. Kildall, G. A., Global Expression Optimization During Compilation, Ph. D. Dissertation, Computer Science Group, University of Washington, Seattle, June 1972.

31. Kildall, G. A., The ALGOL-E Programming System, Internal Report, Mathematics Department, Naval Postgraduate School, Monterey, California, December 1970.

32. Kildall, G. A. and Roberts, A., "ALGOL-E: An Experimental Approach to the Study of Programming Languages," SIGCSE Bulletin, n. 4, p. 127-135, March 1972.

33. Lowery, E. S. and Medlock, C. W., "Object Code Optimization," Communications of the ACM, v. 12, n. 1, p. 13-22, January 1969.

34. Lukasczyk, N., An Investigation of Selection Methods for a Simple Program Flow Analysis Algorithm, Masters Thesis, U. S. Naval Postgraduate School, Monterey, June 1974.

35. Maggiolo-Schettini, A., "Procedure Linkage Optimization Working Paper," Conference Record of the ACM Symposium on Principles of Programming Languages, p. 183-193, Boston, Massachusettes, October 1973.

36. McKeeman, W., "Peephole Optimization," Communications of the ACM, v. 8, n. 7, p. 443-444, July 1965.

37. McKeeman, W. M., Horning, J. J. and Wortman, D. B, A Compiler Generator, Prentice-Hall, Inc., 1970.

38. Naur, P., "Revised Report on the Algorithmic Language ALGOL 60," Communications of the ACM, v. 6, p. 1-17, January 1963.

39. Painter, J. A., "Effectiveness of an Optimizing Compiler for Arithmetic Expressions," SIGPLAN Notices of the ACM, v. 5, n. 7, p. 101-126, July 1970.

40. Rosen, B. K., "Note on Semantics and Optimization," SIGPLAN Notices of the ACM, v. 9, n. 11, p. 6-10, November 1974.

41. Sethi, R. and Ullman, J. D., "Generation of Optimal Code for Arithmetic Expressions," Journal of the ACM, v. 17, n. 4, p. 715-728 October 1970.

42. Tarjan, R. H, "Depth-First Search and Linear Graph Algorithms," SIAM Journal on Computing, v. 1, n. 2, p. 146-160, June 1972.

43. Urschler, G., Complete Redundant Expression Elimination in Flow Diagrams, IBM Thomas J. Watson Research Center Computer Sciences Department RC 4965 (#22020), August 1974.

44. Zelkowitz, M. V. and Bail, W. G., "Optimization of Structured Programs," Software - Practice and Experience, v. 4, p. 51-57, 1971.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Chairman, Code 72 Computer Science Group Naval Postgraduate School Monterey, California 93940	2
4.	Professor Gary A. Kildall, Code 72Kd (advisor) Naval Postgraduate School Monterey, California 93940	1
5.	Professor Daniel L. Davis, Code 53Dv Naval Postgraduate School Monterey, California 93940	1
6.	Ensign Gerald G. Brown USN, Code 55Zr Naval Postgraduate School Monterey, California 93940	1
7.	Lieutenant(JG) Jack W. Cowan USN (student) NAVCOSSACT Washington Navy Yard Washington, D. C. 20390	1

160682
Thesis
C756923 Cowan
c.1
An implementation of an iterative global flow analysis algorithm.